Passing the Torch: Old School Red Teaming, New School Tactics?

Dave McGuire, Will Schroeder
Veris Group's Adaptive Threat Division

@davidpmcguire

- Director of Veris Groups’ Adaptive Threat Division
  - Penetration Testing
  - Red Teaming
  - Threat Replication

- Former technical lead of NSA Red Team

- Developer of **Adaptive Penetration Testing** and **Adaptive Red Team Tactics** BlackHat training classes

@harmj0y

- Chief security researcher and red teamer for Veris Group's Adaptive Threat Division

- Co-founder of the Veil-Framework #avlol
  - www.veil-framework.com
  - **Shmoocon '14:** AV Evasion with the Veil Framework
  - **Defcon '14:** Veil-Pillage: Post-exploitation 2.0
  - co-wrote Veil-Evasion, wrote Veil-Catapult, Veil-Pillage, PowerView, and PowerUp

- Active Cortana and PowerShell hacker

tl;dr

- Pentesting vs. Red Teaming
- Red Teaming vs Red Team Operations

- **Tactic 1:** Situational Awareness
- **Tactic 2:** Domain Trusts
- **Tactic 3:** Escalation and Pivoting
- **Tactic 4:** Persistence
- **Tactic 5:** Files Files Files

- **Demo**: FIGHT!

Pentesting

- Definition ranges anywhere from a single person running a (slightly)-glorified vuln scan, to a full on multi-person assault for several weeks

- **Reasonable Balance:** breadth vs. depth, find as many holes as you can and see how far you can get in a limited timeframe

Red Teaming vs.
Red Team Operations

- Red teaming means different things to different people
- Some focus on physical ops, some focus on in-depth social engineering, some focus on custom exploit dev, some focus on pure network based operations, etc.
- Common thread of increased time frame and more permissive scope

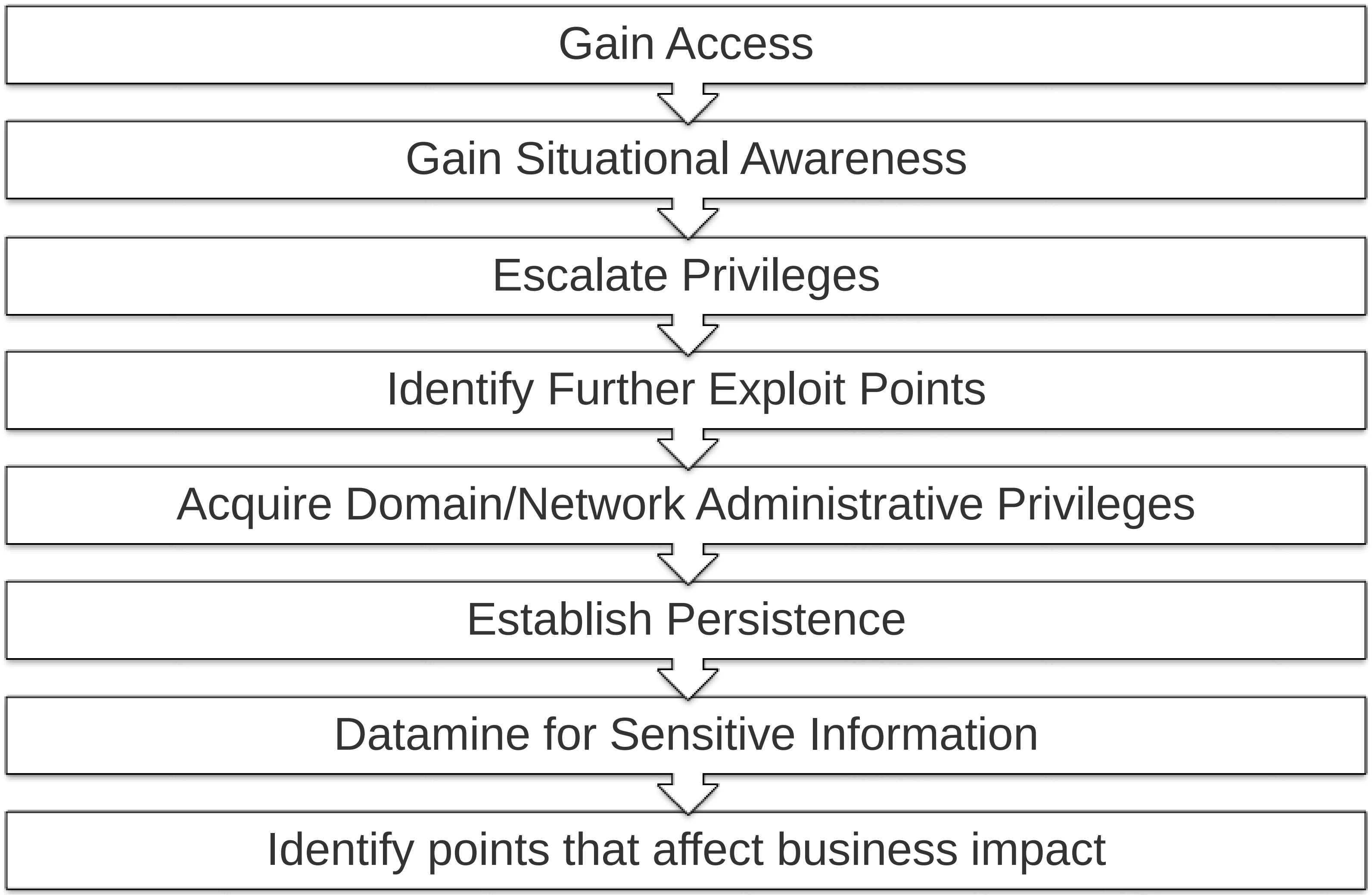
Red Teaming Operations

- An operation organized to emulate a potential adversary's exploitation or attack effectiveness against a targeted mission or capability
- Military concept of adversarial thinking that evolved into adversary emulation
- **General idea:** simulate an "advanced" attacker

Our Take

- We focus on operational risk posed by advanced attackers
- From our perspective, **red team operations primarily involve analysis and actions that happen after the initial access**
- A vast majority of corporate deployments in the US consist of Windows environments, so that’s where we focus

Cyber Kill-Chain :)


Gain Access
Gain Situational Awareness
Escalate Privileges
Identify Further Exploit Points
Acquire Domain/Network Administrative Privileges
Establish Persistence
Datamine for Sensitive Information
Identify points that affect business impact

Bridging the Gap

- Red Teaming is historically defined by:
  - The use of specialized toolsets
  - Expanded timeframe
  - Large team size
  - Lots of $$$

- Our interpretation is really more about emulation of techniques, independent of toolsets
  - Newer tools provide many previously specialized capabilities

Nothing New?

- These techniques are public but lesser known
- Admins need to admin, users wanna use
  - Always going to be a way to abuse ‘normal’ functionality for unintended purposes
- Everything here is possible through multiple means
  - VBscript, PowerShell, C/WinAPI or native/CLI
  - Good to have alternative ways to accomplish the same goal

Tactic 1

Situational Awareness

Landing on the Beachhead

Landing on the Beachhead

- Orient yourself after the initial compromise

- Gain situational awareness to plan your next attack steps

- Nothing revolutionary here:
    - the more information you can gather, the better you can map out your next phase
    - and active directory is a gold mine of information

Old School: Users/Network Info

- Groups/users in the domain:
  - net users /domain
  - net group /domain
  - net group “Domain Admins” /domain
- Computers in the domain:
  - net view /domain:<domain name>
- Information about a host
  - net view \\<hostname>
  - srvinfo \\<hostname>
  - sc \\<hostname>
  - nbtstat -A <hostname>

Old School: User Hunting

- Find where high value users are logged in
- Find user fileservers:
  1. **net use**
     a. look for mapped drives
  2. **net user <username> /domain**
     a. extract “Home Directory” server
  3. ...repeat for all users :(
- Check the sessions, match against target users:
  - **NetSess.exe SERVER**

New School

- Rob Fuller (@mubix’s) netview.exe project, presented at Derbycon 2012, is a tool to “*enumerate systems using WinAPI calls*”

- Finds all machines on the network, enumerates shares, sessions, and logged in users for each host
  - And now can check share access, highlight high value users, and use a delay/jitter :)

New(est) School: PowerShell

- PowerShell has some great AD hooks and access to the Windows API as well
- **PowerView** implements a ton of this functionality without having to remember all the syntax
- Full replacement for “net *” commands, as well as a full netview.exe implementation, **Invoke-Netview**

New(est) School: PowerShell

- **Invoke-UserHunter**
  - queries AD for all machines
  - queries for a target user group (“Domain Admins”)
  - uses the same API calls as netview.exe to enumerate sessions and logged in users, matching against the target user list
- **Invoke-StealthUserHunter**
  - queries AD for all users, extracts all home directories
  - queries for a target user group
  - runs the equivalent to “net session” against each file server, matching against target user list

Tactic 2

Domain Trusts

Domain Trusts

Windows Domain Trusts 101

- Trusts allow separate domains/directories to form inter-domain relationships

- A trust simply allows for the possibility of access between domains
  - Administrators must go the extra mile and actually enable access

- Trusts can be a method for an attacker to jump from one network to another

Domain Trusts 101

- Trusts come in 3 varieties
  - **One way** - Only one domain trusts the other
  - **Two way** - Both domains trust each other
  - **Transitive** - Domain A trusts Domain B and Domain B trusts Domain C, so Domain A trusts Domain C
- Each domain in a forest has a two-way transitive trust with both its parent and each of its children
- More information:
  - http://www.harmj0y.net/blog/redteaming/trusts-you-might-have-missed/

So What?

- ***Why does this matter?***
- Red teams often compromise accounts/machines in a domain trusted by their actual target
    - Allows operators to exploit these existing trust relationships to achieve their end goal
- And **Enterprise Admin** = pwnership over everything below

Old School: nltest

- Some Microsoft administrative tools can give you lots of interesting information concerning domain trusts:
  - **nltest /domain_trusts** - identify all current domain trusts
  - **nltest /dcname:<domain name>** - identify primary domain controller for a target domain
- Other tools grant some of this functionality as well:
  - **netdom** to verify two-way trusts
  - **dsquery/dsget** to enumerate additional information

Old School: dsquery/dsget

- Retrieve users from a specific domain:
  - **dsquery user “cn=users,dc=dev,dc=test,dc=local”**
- Grab “Domain Admins” for a specific domain:
  - **dsget group "cn=Domain Admins,cn=users,dc=dev,dc=test,dc=local" -members**
- See what groups a user is a member of:
  - **dsget user "cn=john,cn=users,dc=dev,dc=test,dc=local" -memberof**

New School: Trusts and PowerShell

- Of course you can do this (and with greater ease) using PowerShell:
  - `([System.DirectoryServices.ActiveDirectory.Forest]::GetCurrentForest()).Domains`
  - `([System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()).GetAllTrustRelationships()`
- PowerShell AD functionality can easily operate on domains to which there's an existing trust
  - finding domain controllers, querying users, enumerating domain groups, etc.

New(est) School: PowerView

- Domain/forest trust relationships can be enumerated through several **PowerView** functions:
    - **Get-NetForest**: information about the current domain forest
    - **Get-NetForestTrusts**: grab all forest trusts
    - **Get-NetForestDomains**: enumerate all domains in the current forest
    - **Get-NetDomainTrusts:** find all current domain trusts, á la nltest

New(est) School: PowerView

- If a trust exists, most functions in **PowerView** can accept a “**-Domain <name>**” flag to operate across a trust:
    - Get-NetDomainControllers
    - Get-NetUser/Get-NetUsers
    - Get-NetComputers/Get-NetFileServers
    - Get-NetGroup/Get-NetGroups
    - Invoke-UserFieldSearch
    - Invoke-Netview
    - Invoke-UserHunter, etc.

Tactic 3

Escalation and Pivoting

Escalation and Pivoting

Moving Beyond the Beachhead

- Now that you’ve mapped out the network, active directory structure and trust relationships, time to see what mischief you can cause
- First step often involves escalating to SYSTEM on your target
- Then grab tokens/passwords/etc. and start your lateral movement

Old School: Escalation

- One of the most effective escalation vectors was (and still is) vulnerable Windows services

- Specifically, many organizations overlook the permissions for service binaries :)

- After gaining SYSTEM on a box, makes it a lot easier to snarf up all active user tokens
    - File servers are great places to look

Old School: Tokens

- **Impersonation tokens**
  - for “non-interactive” logons, i.e. drive mapping
  - allows a process/thread to carry out actions as the identified user on the current system
- **Delegation tokens**
  - for “interactive” logons (we want these!!)
  - allows a process/thread to carry out actions as the identified user on remote systems
- Impersonate/steal tokens with your agent of choice
  - Can also just migrate to a user-owned process!

New School: Escalation

- **PowerUp**: a PowerShell tool to automate the discovery and abuse of Windows privilege escalation vectors. Checks for:
  - vulnerable services
  - service binaries
  - unquoted paths
  - AlwaysInstallElevated, and more

- **Invoke-AllChecks** will run all current checks, and will tell you what function will abuse whatever vulns are found

New School: Token Manipulation

- **PowerSploit / Exfiltration / Invoke-TokenManipulation.ps1**
- Equivalent to Incognito’s functionality, but purely in PowerShell
- Allows you to enumerate tokens, steal/impersonate what you find, create processes, etc.

New School: Mimikatz FTW

- If you don’t know what Mimikatz is, shame on you!

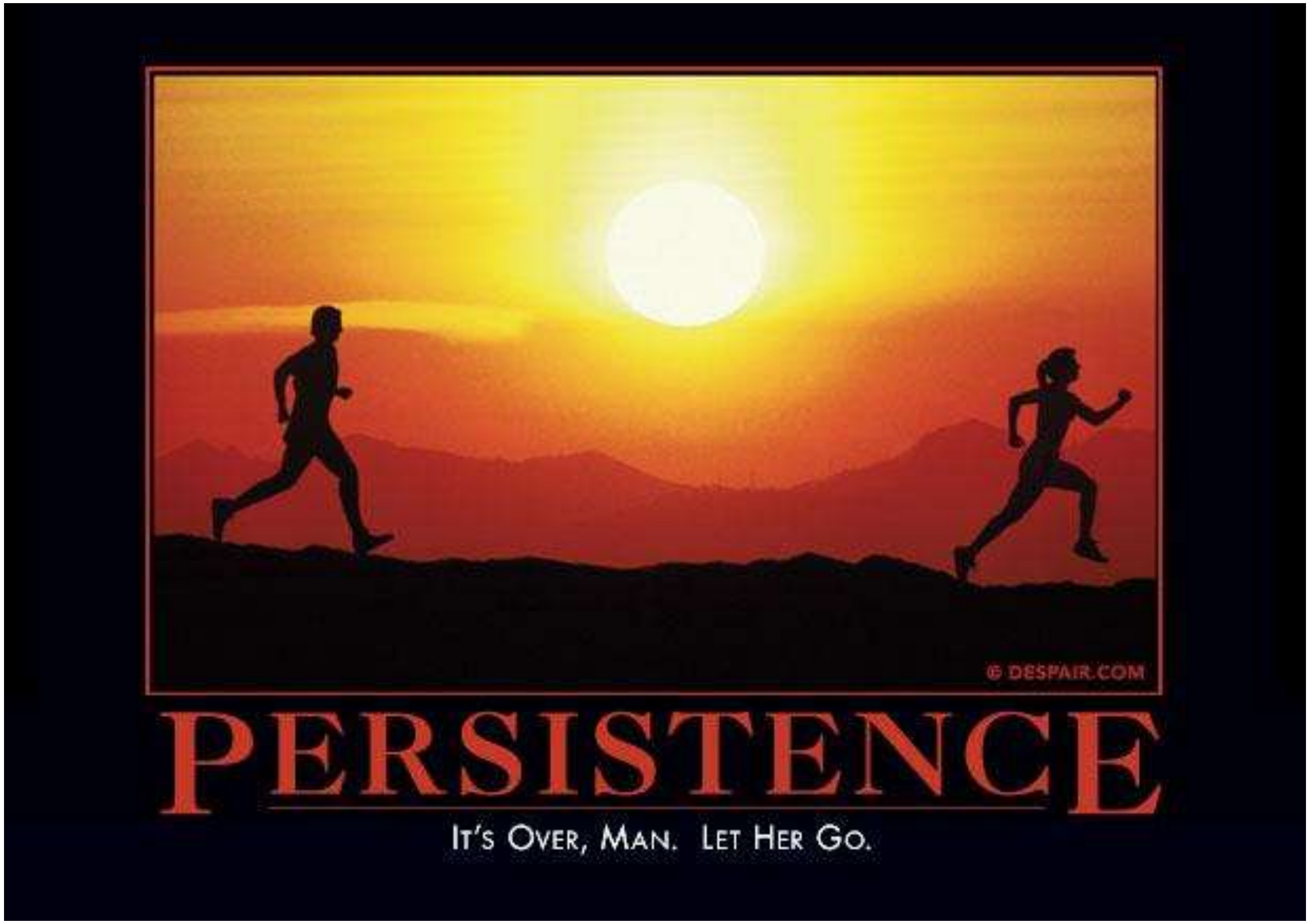
- Even better, the PowerSploit guys integrated everything into PowerShell

- **Invoke-Mimikatz.ps1** lets you dump credentials, play with kerberos tickets, and more

Tactic 4

Persistence

Persistence



Keeping the Door Open

- You don’t want to have to regain access for each new engagement
  - Attackers don’t leave, why should you
  - Long term assessments require stable access
- Two main approaches:
  - local machine level persistence
  - domain level persistence
- Credentials are always a great persistence method

Old School

- Keep a low and slow C2 agent running on the machine
  - In case of reboot, drop an obfuscated binary to disk
- Try to stay off of main servers
  - Find privileged users with access to those servers, then target their workstations
- Dump domain hashes
  - Pay attention to privileged accounts with an infrequent password change policy

New School: Local Persistence

- For low-and-slow agent persistence, a few specialized tools are available:
  - Cobalt Strike’s Beacon
  - Immunity’s Innuendo
  - Silentbreak’s Throwback
- For on-disk local persistence, some (newer?) techniques:
  - schtasks + PowerShell through a one-liner
  - permanent WMI + PowerShell through PowerSploit
  - obfuscated binary + SC

New(est) School: Domain Persistence

- There’s nothing better than...

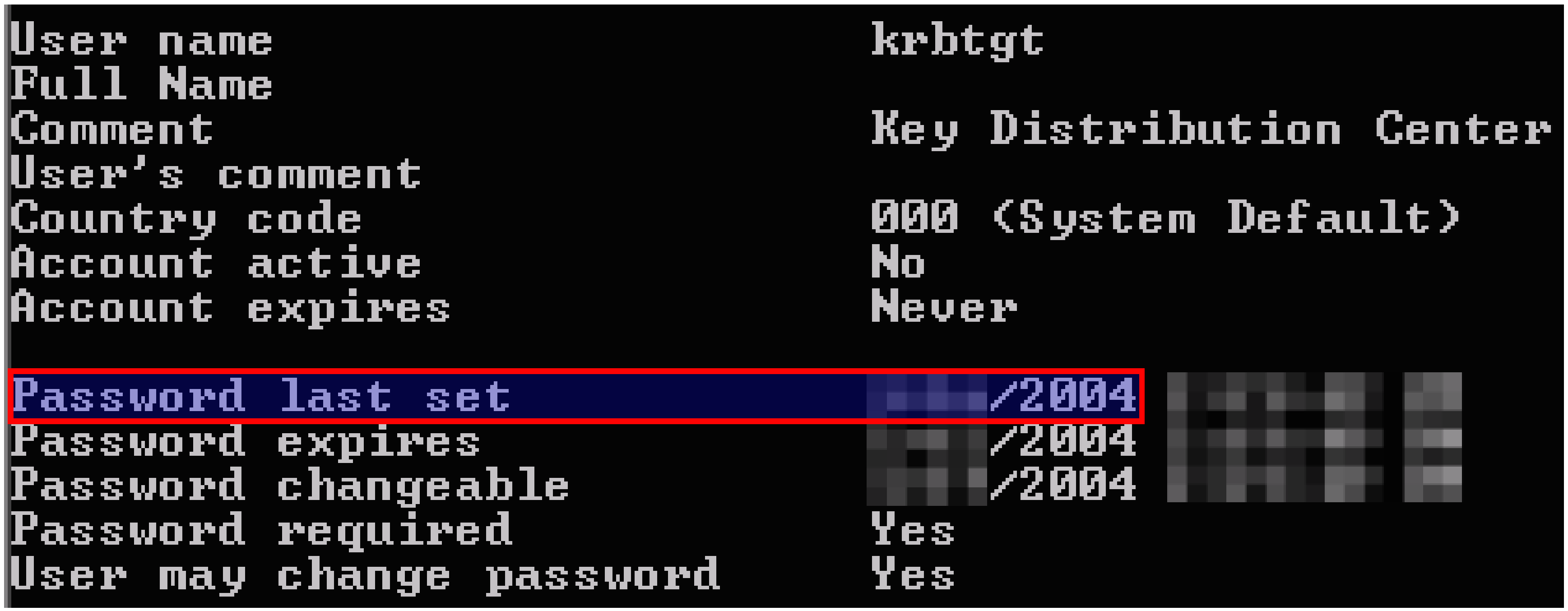
The Golden Ticket

- If you can knock over a domain controller and grab the krbtgt hash, you can forge your own kerberos tickets
  - For any user. And put them in any group. For as long as that hash isn't changed. Which isn't often.
  - Think years.

- Long story short: ***if you can pwn a domain once, you can pwn it for a LONG time***

- Go see Chris’ “***Et tu – Kerberos?***” at 6pm!

A LOOONNNGGG Time



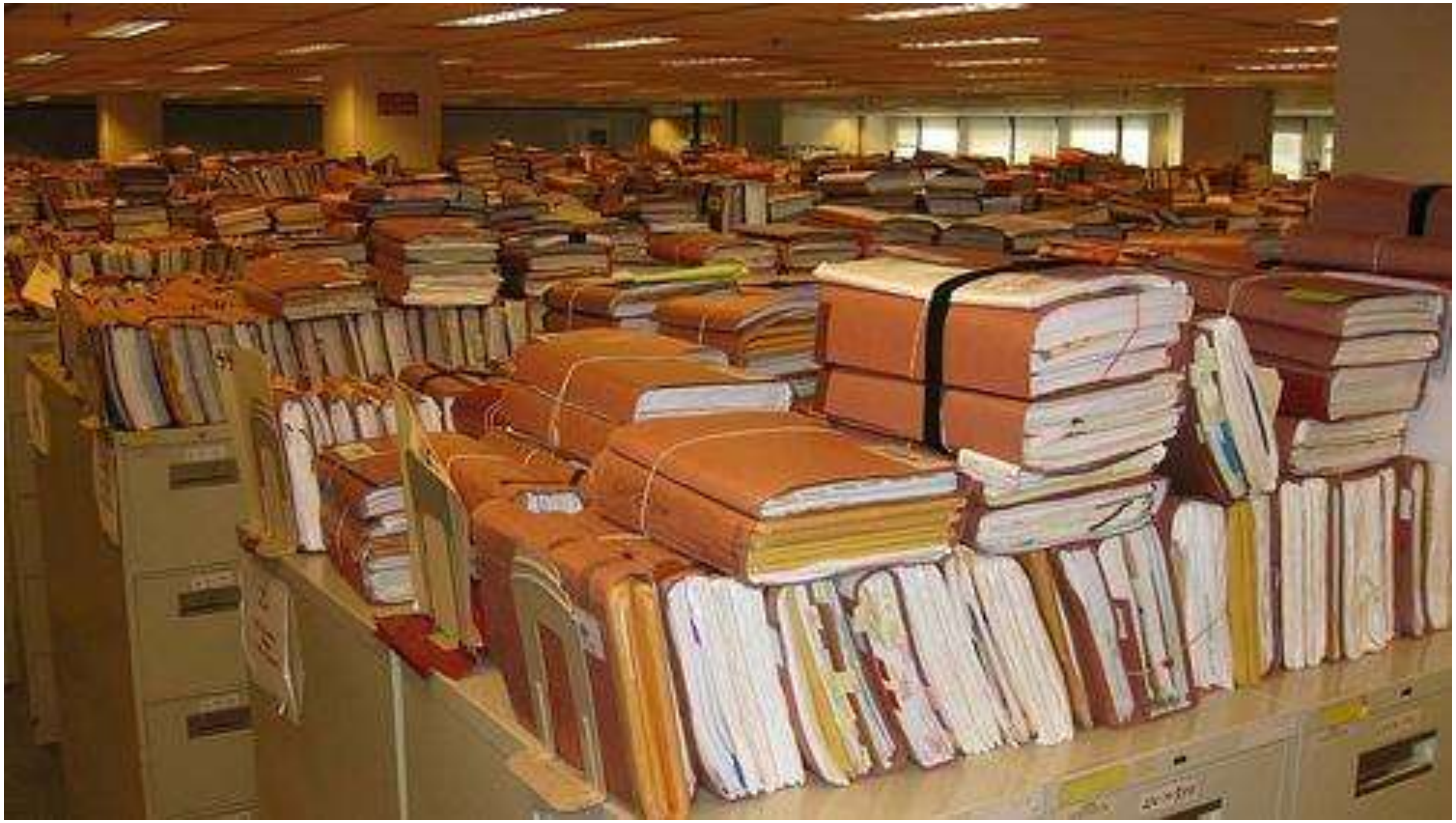
```
User name                    krbtgt
Full Name
Comment                      Key Distribution Center
User's comment
Country code                 000 (System Default)
Account active               No
Account expires              Never

Password last set            [redacted]/2004 [redacted]
Password expires             [redacted]/2004 [redacted]
Password changeable          [redacted]/2004 [redacted]
Password required            Yes
User may change password     Yes
```

Tactic 5

Files Files Files

Files on Files



Files on Files

- **The end goal isn’t domain admin, the end goal is data**
- Even when you don’t own everything, every organization has file shares with improper access controls
- **Goal:**
  - Locate and triage every single file we can access over the network (hunt for sensitive data)
  - Gain sensitive information (potentially for escalation), choose possible files to trojanize

Old School: Finding Shares

- Finding shares manually:
  - **net view /domain:<domain name>**
  - **net view \\<hostname>**
- Make new people triage every network share and file found
  - Great morale booster :)
- Once readable shares are located, triage the possible thousands to millions of files on remote servers
  - New people like doing this :)

Old School: Finding Files

- Nothing more old school that straight recursive directory listings with **dir /s**
    - **dir /s \\<hostname>\SHARE > listing.txt**
    - **dir /s /Q /O:-D /T:A \\<hostname>\SHARE > listing.txt**
- Then grep file listings for sensitive names, as well as office docs/.exe’s that have been recently accessed :)

New School: Finding Shares

- Search for open shares and sensitive files with PowerShell and **PowerView**

- **Invoke-ShareFinder -CheckAccess** will:
  - Find all machines on the network
  - Enumerate all shares on each machine
  - Check if the current user has read access to any found shares

- Spits out a “\\HOST\SHARE - comment” list of all shares on the network you can read

New School: Finding Files

- Once you have shares, PowerShell helps you triage for files, nicely sortable:

    -
      ```
      PS> get-childitem \\MACHINE\PATH -rec -ErrorAction SilentlyContinue | where {!$_.PSIsContainer} | select-object FullName,@{Name='Owner';Expression={(Get-Acl $_.FullName).Owner}}, LastAccessTime, LastWriteTime, Length | export-csv -notypeinformation -path files.csv
      ```
- The **-include @("*term*")** argument lets you find files by wildcard terms

New School: Targeted Trojanation

- **Invoke-SearchFiles** and **Invoke-FileFinder** both accept the "**-FreshEXEs**" flag
  - this will find .exe's accessed within the last week
- We can then use Joshua Pitts' **The Backdoor Factory** to easily trojanate these binaries
- Then can use **PowerView**'s **Invoke-CopyFile** to copy the trojanated file in, matching MAC attributes

Demo

PLAYER 1
K.O.
99
CPU
Ryu
Blanka
FIGHT
COMBO
REVENGE
COMBO
REVENGE

Recap

- Newer tools and techniques can greatly facilitate red team engagements
- Always have a backup plan- if one implementation fails, you always need to have options
- **Moral of the story**: the underlying tactics rarely change, but the specific implementations often do

Questions?

- Offensive PowerShell blogs:
    - http://obscuresecurity.blogspot.com/
    - http://www.exploit-monday.com/
    - http://www.darkoperator.com/blog/
    - http://blog.harmj0y.net

- Offensive PowerShell Toolsets:
    - **PowerSploit:**
        - https://github.com/mattifestation/PowerSploit/
    - **PowerView:**
        - https://github.com/veil-framework/Veil-PowerView
    - **PowerUp:**
        - https://github.com/HarmJ0y/PowerUp