CrowdsourcedDFIRBook

A crowdsourced DFIR book by members of the Digital Forensics Discord Server

Andrew Rathbun, ApexPredator, Kevin Pagano, Nisarg Suthar and John Haynes

CrowdsourcedDFIRBook

A crowdsourced DFIR book by members of the Digital Forensics Discord Server

Andrew Rathbun, ApexPredator, Kevin Pagano, Nisarg Suthar and John Haynes

This book is for sale at http://leanpub.com/crowdsourceddfirbook

This version was published on 2022-05-23

ISBN 979-8-9863359-0-2

Contents

Chapter 0 - Introduction

TODO for Andrew

Authors

Author bios go here

Contributors

- brootware - thank you for helping with the dead link checker[1]!

[1] https://github.com/Digital-Forensics-Discord-Server/CrowdsourcedDFIRBook/issues/59

Section 1 - General

Chapter 1 - History of the Digital Forensics Discord Server

Special thanks to Kevin Pagano for creating the logo for the Digital Forensics Discord Server!

History of the Digital Forensics Discord Server

I felt it was prudent to choose this topic for this project because very few others could provide as in depth of an account on the history of the Digital Forensics Discord Server. More to come in this section.

Beginnings in IRC

Long before the Digital Forensics Discord Server came to be, there existed a channel on an IRC[2] network called freenode[3]. The channel was called `#mobileforensics`. This channel had its humble beginnings on a Google Group ran by Bob Elder of TeelTech[4], called the Physical and RAW Mobile Forensics Group[5], which still exists today. In order to gain access to this Google Group, one had to have attended a TeelTech training in the past. It was, and continues to be, a phenomenal resource for those of us in Law Enforcement trying to navigate the waters of mobile forensic acquisitions.

By way of background, In February 2016 I attended the JTAG/Chip-Off class by TeelTech taught by Mike Boettcher and gained an invite to the Physical and RAW Mobile Forensics Group. I actively participated in the group to the extent my knowledge and curiosity enabled me. Make no mistake about, almost every other active poster in that group was more experienced or knowledgeable than myself. However, I thought to myself that there was no better place to immerse myself in or people to surround myself with than this group if I wanted to be the best version of myself.

On August 23, 2016, a user that went by the name of `tupperwarez` had informed the group that they were starting an IRC channel

[2] https://en.wikipedia.org/wiki/Internet_Relay_Chat
[3] https://en.wikipedia.org/wiki/Freenode
[4] https://www.teeltech.com/
[5] https://groups.google.com/g/physical-mobile-forensics/about?pli=1

called `#mobileforensics` in an effort "exchange ideas & have live discussions", as the post stated. I have been using forums for all of my internet life up until this point and I think subconsciously I was ready for something more, and this was it! I also knew that IRC was a longstanding tradition but I had never dabbled with it as I only had previous experience with messaging clients such as AOL Instant Messenger (AIM)[6] and MSN Messenger[7] at the time. 13 minutes after the post went out by `tupperwarez`, I was the first to respond in the thread that I had joined.

Throughout the next year and a half, a small contingent of people totaling anywhere from 7-15 at any given time occupied this IRC channel. We became a really tightknit group of examiners who relied on each other's knowledge and expertise to navigate challenges in our everyday casework. These problems often would relate to performing advanced acquisition methods using Chip-Off, JTAG, or flasher boxes. The collaboration was exactly what I was looking for because through each other we were able to cast a wider net for knowledge that we sought for problems we were coming across in our everyday investigations.

I recall utilizing an application called HexChat[8] to access this IRC channel. I'd have HexChat open at all times along with my everyday workflow of software applications to perform my duties as a Detective. For those reading this who have not used IRC before, know that's its nowhere near as feature rich as Discord. Discord is much more modern and IRC has been around since the "early days" of the internet as we know it today. I bring this up because often we needed to share pictures with each other as an exhibit for a problem we were encountering during the acquisition or decoding process of a mobile device.

[6] https://en.wikipedia.org/wiki/AIM_(software)
[7] https://en.wikipedia.org/wiki/Windows_Live_Messenger
[8] https://hexchat.github.io/

Move to Discord

Truthfully, I had forgotten this detail I'm about to share but one of the moderators reminded me of it a couple of years ago and it all came back to me. One of the main catalysts for moving from IRC was the fact that I was really annoyed with having to upload a picture to imgur and share the link on the IRC channel as it seemed inefficient and the process grew stale for me. I had created a Discord account back in September 2016 to join various special interest servers so I had a fair amount of exposure to Discord's capabilities prior to the birthdate of the Digital Forensics Discord Server, which is March 26th, 2018.

I recall having aspirations for a move to Discord months prior to March 2018. For those who didn't use Discord around this time, it was primarily a platform marketed towards gamers. Using it for things other than gaming wasn't the intended purpose at the time, but the functionality it had was everything I wanted in a chat client. Take all of the good features from every other chat application I had used up until that point in time and add even more quality of life features and an awesome mobile application, and I was sold. I didn't like how it wasn't as seamless to use IRC on my phone and combined with the inefficient image uploading process, Discord was a breath of fresh air.

I was reminded that the major push to move to Discord came from me mostly surrounding the image uploading process combined with the positive experiences I had with the platform in my personal life via special interest servers from September 2016 to March 2018. The call to move to Discord was met with nearly unanimous approval from members of the IRC channel. As a result, the Mobile Forensics Discord Server was created!

Mobile Forensics Discord Server -> Digital Forensics Discord Server

The Mobile Forensics Discord Server enjoyed great success and rapid growth throughout its first year of existence. The server's growth was entirely driven by word of mouth and advertising on various Google Groups. The list of channels maintained in the server were driven by member requests which quickly expanded outside of mobile devices. Over time, it became increasingly apparent that branding the server as a Mobile Forensics server did not fully encompass the needs of the DFIR community. To the best of my research, the Mobile Forensics Discord Server was rebranded to the Digital Forensics Discord Server sometime around February 2019.

Since then, multiple channels have been added, renamed, and removed at the request of members.

Member Growth

Throughout the 4 years (as of this writing), the Digital Forensics Discord Server has undergone substantial growth. Below are some major membership milestones that were mined from Announcements I made in the `#announcements` channel over time.

Major Milestones

Date	# of Members
3/26/2018	3
3/29/2018	116
4/3/2018	142
4/6/2018	171
4/11/2018	200
4/13/2018	250
5/30/2018	300
6/28/2018	375
7/9/2018	400
7/25/2018	450
8/20/2018	500
9/27/2018	600
11/16/2018	700
12/6/2018	800
1/10/2019	900
2/1/2019	1000
5/8/2019	1500
10/4/2019	2000
1/30/2020	2500
3/27/2020	3000
5/22/2020	4000
3/26/2021	6800
8/2/2021	8000
1/29/2022	9000
3/26/2022	9500

Hosting the Magnet Virtual Summit 2020

In early 2020, shortly after the COVID-19 pandemic began, I was approached by representatives from Magnet Forensics inquiring about the possibility of providing a centralized location for attendees of the Magnet Virtual Summit 2020 to chat during presentations. Enthusiastically, we accepted the idea and began to plan the logistics of hosting what likely would become a large influx of members. I seem to recall nearly 1500 members joining during the month long Magnet Virtual Summit 2020.

In retrospect, it's clear that this was one of the first indicators that the server had "made it" in the eyes of the community.

SANS DFIR Virtual Summit 2020 went virtual

Community Engagement Within the Server

vendors and customers

Impact on the DFIR community

solo cop up in alaska who's nearest fellow examiner is 3 hours away

Law Enforcement roles were separated by country from the early stages of the server for the purpose of delineating members from each other due to various legal considerations that may vary from one jurisdiction to another. Because of that, enumerating a list of the countries that a Law Enforcement role has been created for is likely the best way to establish the reach the Digital Forensics Discord Server has had on the DFIR community on a global level.

Countries with roles assigned for Law Enforcement personnel (as of May 2022):

- Albania
- Argentina
- Australia
- Austria
- Bangladesh
- Belgium
- Bosnia
- Brazil
- Canada
- Chile
- China

- Columbia
- Croatia
- Cyprus
- Czech Republic
- Denmark
- Dominican Republic
- Estonia
- Finland
- France
- Germany
- Greece
- Grenada
- Iceland
- India
- Indonesia
- Iraq
- Ireland
- Israel
- Italy
- Jamaica
- Japan
- Korea
- Latvia
- Lithuania
- Luxembourg
- Maldives
- Malaysia
- Malta
- Mongolia
- New Zealand
- Mauritius
- Mexico
- Monaco
- Nepal

- Nigeria
- Norway
- Pakistan
- Netherlands
- Poland
- Portugal
- Romania
- Royal Cayman Islands
- Russia
- Senegal
- Seychelles
- Singapore
- Slovakia
- Slovenia
- Spain
- Sweden
- Switzerland
- Taiwan
- Turkey
- United Arab Emirates
- United Kingdom
- Ukraine
- Uruguay
- USA
- Vietnam

To save you from counting, that's 70 countries with a dedicated Law Enforcement role. This means that someone who has identified themselves as someone who works in Law Enforcement in one of these countries has joined the server and had this role assigned to them. At least 1 person from each of these countries that at the time served in a Law Enforcement capacity have joined the Digital Forensics Discord Server. With 195 countries[9] recognized in the

[9] https://www.worldatlas.com/articles/how-many-countries-are-in-the-world.html

world as of the writing of this book, the server has a reach into approximately 36% of those!

Future

The Digital Forensics Discord Server will continue to live and thrive so long as the community wills it.

For those who are new to administering Discord servers, one important thing to know is that only the member who is assigned as the Server Owner can delete the server. Currently, that person is me, Andrew Rathbun. In the interest of ensuring the Digital Forensics Discord Server lives far beyond all of us (assuming Discord is still around by that time), I've established a paper trail for any other moderators to follow should anything happen to me to where I will never be able to log back in to Discord. This paper trail will require a lot of effort and coordination with family members/friends of mine to access my password vault and many other necessary items in order to Transfer Ownership[10] so that the server can live on without any administrative hiccups.

[10] https://support.discord.com/hc/en-us/articles/216273938-How-do-I-transfer-server-ownership

Chapter 2 - Most Common Data Stores in Mobile Forensics

By Alexis Brignoni[11]

The Artisanal Approach

Most mobile forenisc examinations involve the use of third party tools to extract and decode information stored whitin targeted devices. What happens when the tool presents little to nothing of what is expected? What to do when the targeted app seems to not exist as far as the tool is concerned?

A big part of digital forensics involves what I call The Artisanal Approach. The Oxford Languages dictionary defines artisanal as:

ar·ti·san·al
/ärˈtēzən(ə)l/
adjective
- relating to or characteristic of an artisan.

[11] https://linqapp.com/abrignoni

"artisanal skills"
- (of a product, especially food or drink) made in a traditional or non-mechanized way.
"artisanal cheeses"

This is just a long way of saying that we will rely have to manually identify the relevant data stores. The approach has 3 steps.

1. Locate the relevant apps.
2. Identify the data stores for the app and extract meaningful items.
3. Report generation.

On this chapter we will focus mostly on step number two. We will discuss what type of data stores are mostly seen in mobile forensics and suggest cost effective (i.e. cheap) solutions to traige these sources. Let's dive in.

Locate Relevant Apps

The mobile forensics world is divided, mainly, between two dominant operating systems. These are Google's Android and Apple's iOS operating systems. As such both will organize things in vastly different ways within their file systems. This chapter will present examples from a full file system extraction view of Android and iOS devices. Even when working from a different type of extraction the main concepts, and the handling of data stores, will be the same. For details on mobile extraction types and their differences see here: https://privacyinternational.org/long-read/3256/technical-look-phone-extraction[12]

Is is important to note that this chapter will touch on the most common locations and types of data needed for analysis. It is not

[12]https://privacyinternational.org/long-read/3256/technical-look-phone-extraction

an all encompasing guide to mobile forensics nor does it intend to be so. Without further ado let's dive in.

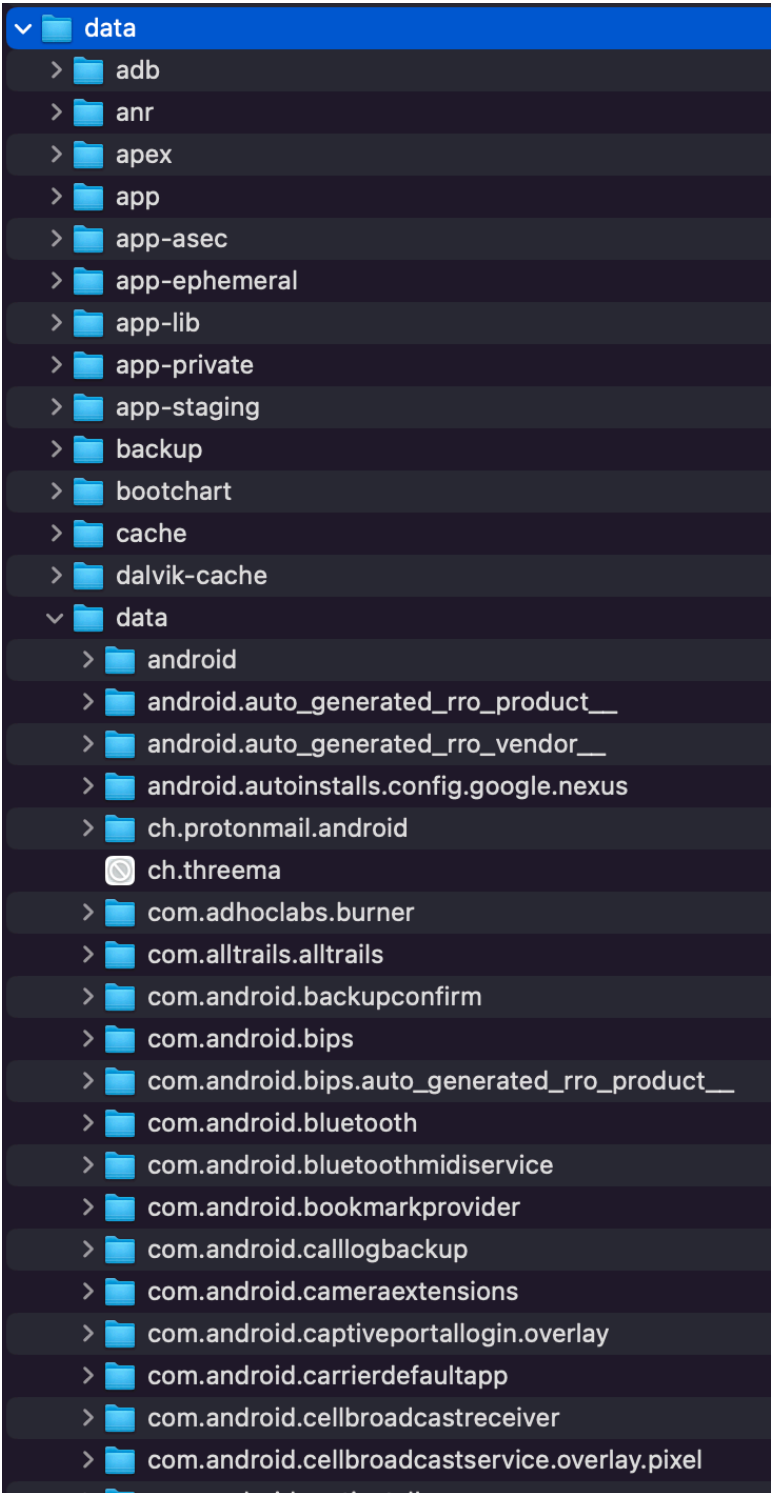
Relevant Apps in Android

In Adroid devices the apps keep most user generated data in the following directory:

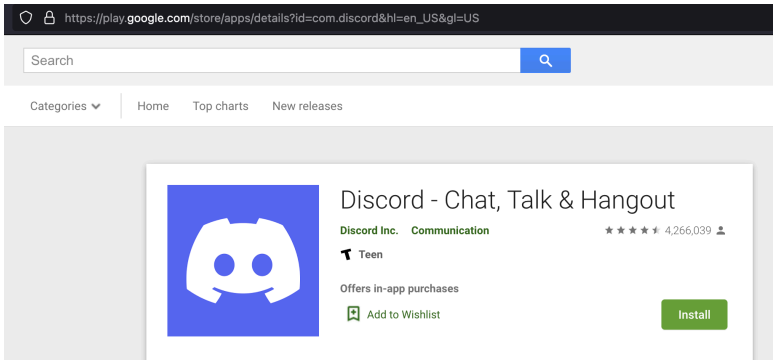
/data/data/

As seen in figure 2.1 there are folders within the data directory for each app on the device. These folders are named in reverse URL format and are known as bundle identifiers (IDs.) For details on bundle IDs in Android see here: https://developer.android.com/studio/build/configure-app-module [13]

[13] https://developer.android.com/studio/build/configure-app-module


data
adb
anr
apex
app
app-asec
app-ephemeral
app-lib
app-private
app-staging
backup
bootchart
cache
dalvik-cache
data
android
android.auto_generated_rro_product__
android.auto_generated_rro_vendor__
android.autoinstalls.config.google.nexus
ch.protonmail.android
ch.threema
com.adhoclabs.burner
com.alltrails.alltrails
com.android.backupconfirm
com.android.bips
com.android.bips.auto_generated_rro_product__
com.android.bluetooth
com.android.bluetoothmidiservice
com.android.bookmarkprovider
com.android.calllogbackup
com.android.cameraextensions
com.android.captiveportallogin.overlay
com.android.carrierdefaultapp
com.android.cellbroadcastreceiver
com.android.cellbroadcastservice.overlay.pixel

Most mobile apps have bundle ID names that are easy to identify. Notice in the previous image how it is pretty obvious that com.android.chrome should be the bundle ID for the Chrome Browser, which it is. Another example would be how the bundle ID for Discord is com.Discord. Be aware that is not always the case. Not all bundle ID names are easy to reference back to the app name just by reading. One way of determining the budle ID of an app in Android is to look for the app in the Google Play store using a browser.

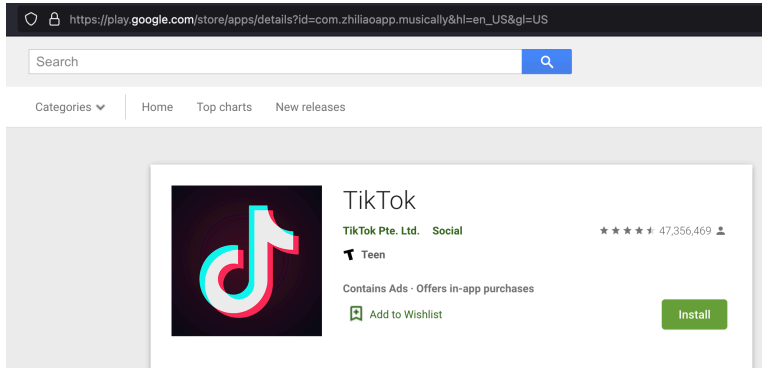


2.2 -

The bundle ID is located in the URL at the top of the page.

https://play.google.com/store/apps/details?id=com.discord&hl=en_-US&gl=US

Let's look at TikTok.



2.3 -

Notice how the bundle ID for TikTok, com.zhiliaoapp.musically makes no obvious reference to TikTok at all.

https://play.google.com/store/apps/details?id=com.zhiliaoapp.musically&hl=en_-US&gl=US

By changing the Google Play store URLs to the possibly unknown bundle IDs found on the target extraction one can determine the common app name for it.

It is of note that apps also save data to additional locations within the Android device. Look for targeted bundle ID direcotries in the following locations:

/data/media/
/MNT/ or /NONAME/

As you navigate the contents of these directories you will find relevant stores that can contain user generated data.

Relevant apps in iOS

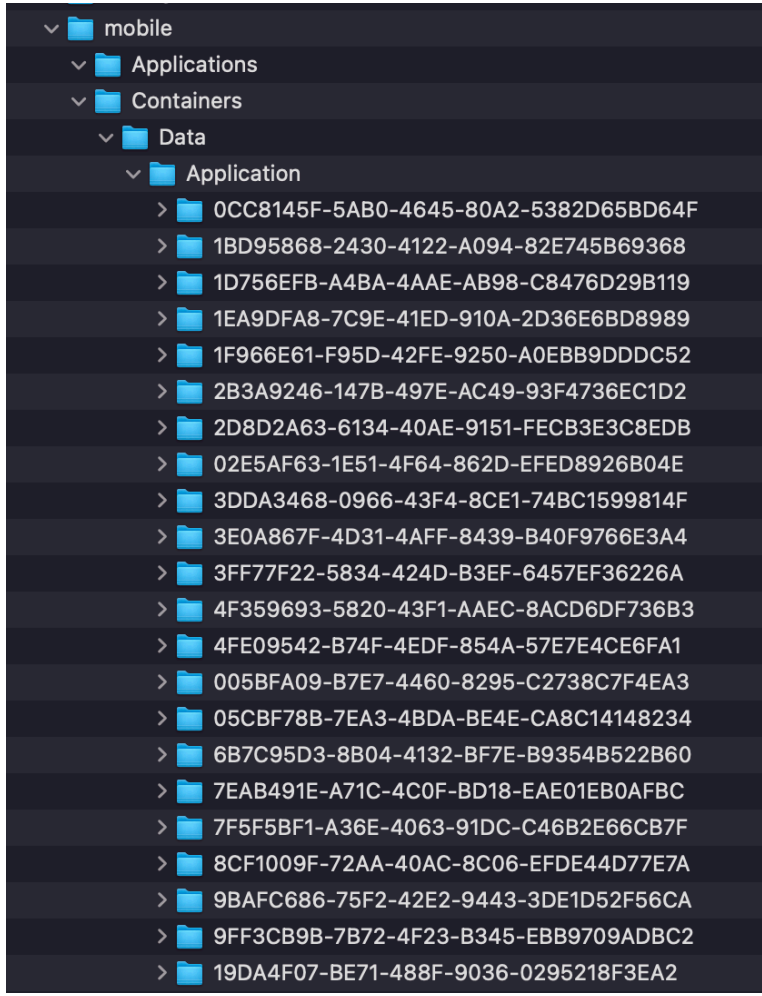
In iOS devices user generated data can be found in the following directories:

/private/var/containers/Bundle/Application/*GUID directory*/
/private/var/mobile/Containers/Shared/AppGroup/*GUID directory*/
/private/var/mobile/Containers/Data/PluginKitPlugin/*GUID directory*/

Notice the *GUID directory* at the end of the paths. These will be subsituted with a corresponding global unique identifier. See the following example:

A72DDBEE-8EEE-4868-9E5A-769B078781EA

These values can change constantly due to app installs, updates, and uninstalls. Unlike Android devices iOS bundle IDs are not part of the application directory paths therefore it is impossible to identify applications of interest visually by bundle ID names.



2.4 -

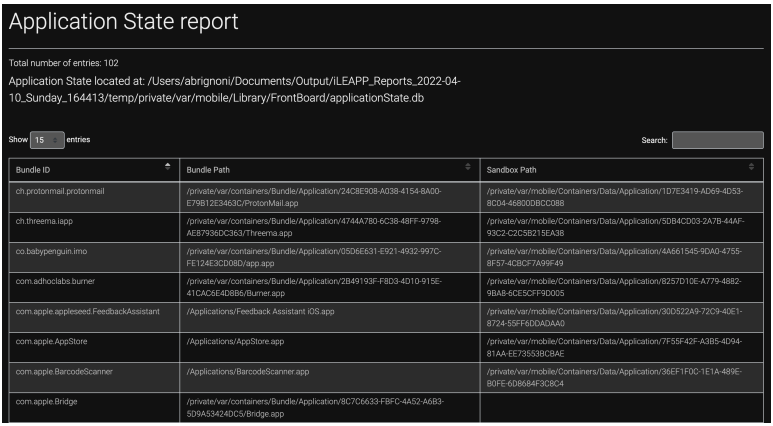
How can then we take these GUID named directories and linked them to the corresponding bundle IDs?

Option 1: applicationState.db

This data store is a SQLite database that contains information for all currently installed appications. We will discuss SQLite databases in the next section. The database is located here:

/private/var/mobile/Library/FrontBoard/applicationState.db

The exemplar data in the following image is from Josh Hickman's test iOS images. These can be found here: https://thebinaryhick.blog/public_images/



Application State report

Total number of entries: 102

Application State located at: /Users/abrignoni/Documents/Output/iLEAPP_Reports_2022-04-10_Sunday_164413/temp/private/var/mobile/Library/FrontBoard/applicationState.db

Show 15 entries Search:

Bundle ID	Bundle Path	Sandbox Path
ch.protonmail.protonmail	/private/var/containers/Bundle/Application/24C8E908-A038-4154-8A00-E79B12E3463C/ProtonMail.app	/private/var/mobile/Containers/Data/Application/1D7E3419-AD69-4D53-8CD4-46800DBCC088
ch.threema.iapp	/private/var/containers/Bundle/Application/4744A7B0-6C38-48FF-9798-AE87936DC363/Threema.app	/private/var/mobile/Containers/Data/Application/5DB4CD03-2A7B-44AF-93C2-C2C5B215EA38
co.babypenguin.imo	/private/var/containers/Bundle/Application/05D6E631-E921-4932-997C-FE124E3CD08D/app.app	/private/var/mobile/Containers/Data/Application/4A661545-9DA0-4755-8F57-4CBCF7A99F49
com.adhoclabs.burner	/private/var/containers/Bundle/Application/2B49193F-F8D3-4D10-915E-41CAC6E4D8B6/Burner.app	/private/var/mobile/Containers/Data/Application/8257D10E-A779-4882-9BA8-6CE5CFF9D005
com.apple.appleseed.FeedbackAssistant	/Applications/Feedback Assistant iOS.app	/private/var/mobile/Containers/Data/Application/30D522A9-72C9-40E1-8724-55FF6DDADAA0
com.apple.AppStore	/Applications/AppStore.app	/private/var/mobile/Containers/Data/Application/7F55F42F-A3B5-4D94-81AA-EE73553BCBAE
com.apple.BarcodeScanner	/Applications/BarcodeScanner.app	/private/var/mobile/Containers/Data/Application/36EF1F00-1E1A-489E-80FE-6D8684F3CBC4
com.apple.Bridge	/private/var/containers/Bundle/Application/8C7C6633-FBFC-4A52-A6B3-5D9A53424DC5/Bridge.app	

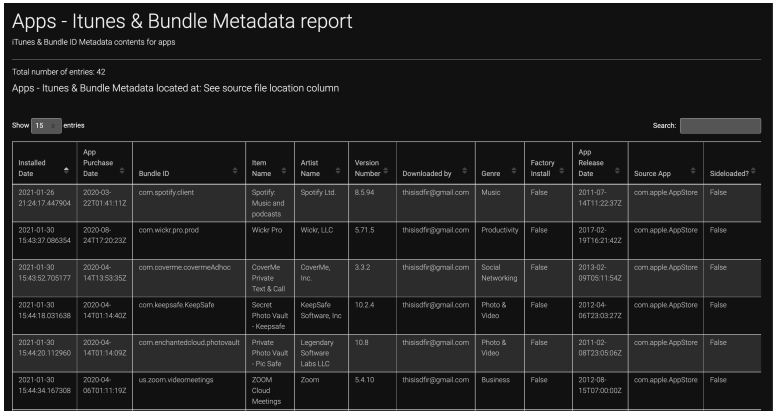
2.5 -

The tool used for this output is iLEAPP and can be found here: https://github.com/abrignoni/iLEAPP

Notice how the database can provide the bundle ID, the app name, and the corresponding GUID values within the paths.

Option 2: iTunesMetadata.plist & BundleMetadata.plist

These data stores are property lists (plist) and will be discussed in the next section. The files reside in each /private/var/containers/Bundle/Application/*GUID directory*/ folder per app. It contains a wealth of information regarding the app for each folder.



Apps - Itunes & Bundle Metadata report

iTunes & Bundle ID Metadata contents for apps

Total number of entries: 42

Apps - Itunes & Bundle Metadata located at: See source file location column

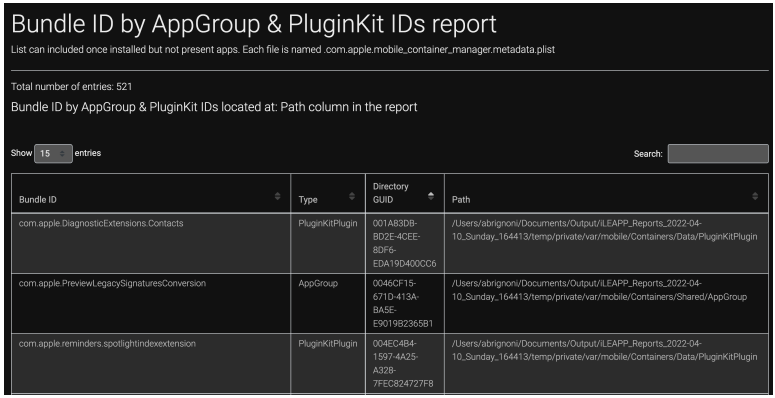
Show 15 entries Search:

Installed Date	App Purchase Date	Bundle ID	Item Name	Artist Name	Version Number	Downloaded by	Genre	Factory Install	App Release Date	Source App	Sideloaded?
2021-01-26 21:24:17.447904	2020-03-22T01:41:11Z	com.spotify.client	Spotify Music and podcasts	Spotify Ltd.	8.5.94	thisisdfir@gmail.com	Music	False	2011-07-14T11:22:37Z	com.apple.AppStore	False
2021-01-30 15:43:37.086354	2020-08-24T17:20:23Z	com.wickr.pro.prod	Wickr Pro	Wickr, LLC	5.71.5	thisisdfir@gmail.com	Productivity	False	2017-02-19T16:21:42Z	com.apple.AppStore	False
2021-01-30 15:43:52.705177	2020-04-14T13:53:35Z	com.coverme.covermeAdhoc	CoverMe Private Text & Call	CoverMe, Inc.	3.3.2	thisisdfir@gmail.com	Social Networking	False	2013-02-09T05:11:54Z	com.apple.AppStore	False
2021-01-30 15:44:18.031638	2020-04-14T01:14:40Z	com.keepsafe.KeepSafe	Secret Photo Vault - Keepsafe	KeepSafe Software, Inc	10.2.4	thisisdfir@gmail.com	Photo & Video	False	2012-04-06T23:03:27Z	com.apple.AppStore	False
2021-01-30 15:44:20.112960	2020-04-14T01:14:09Z	com.enchantedcloud.photovault	Private Photo Vault - Pic Safe	Legendary Software Labs LLC	10.8	thisisdfir@gmail.com	Photo & Video	False	2011-02-08T23:05:06Z	com.apple.AppStore	False
2021-01-30 15:44:34.167308	2020-04-06T01:11:19Z	us.zoom.videomeetings	ZOOM Cloud Meetings	Zoom	5.4.10	thisisdfir@gmail.com	Business	False	2012-08-15T07:00:00Z	com.apple.AppStore	False

2.6 -

Option3: .com.apple.mobile_container_manager.metadata.plist

Like the previous option the data store is a plist. The plist is contained in the /private/var/mobile/Containers/Shared/AppGroup/*GUID directory*/ and /private/var/mobile/Containers/Data/PluginKitPlugin/*GUID directory*/ folders. Notice the period at the start of the plist filename. If using a macOS for analisys make sure to enable the view hidden files options in order to not miss them.



Bundle ID by AppGroup & PluginKit IDs report

List can included once installed but not present apps. Each file is named .com.apple.mobile_container_manager.metadata.plist

Total number of entries: 521

Bundle ID by AppGroup & PluginKit IDs located at: Path column in the report

Show 15 entries Search:

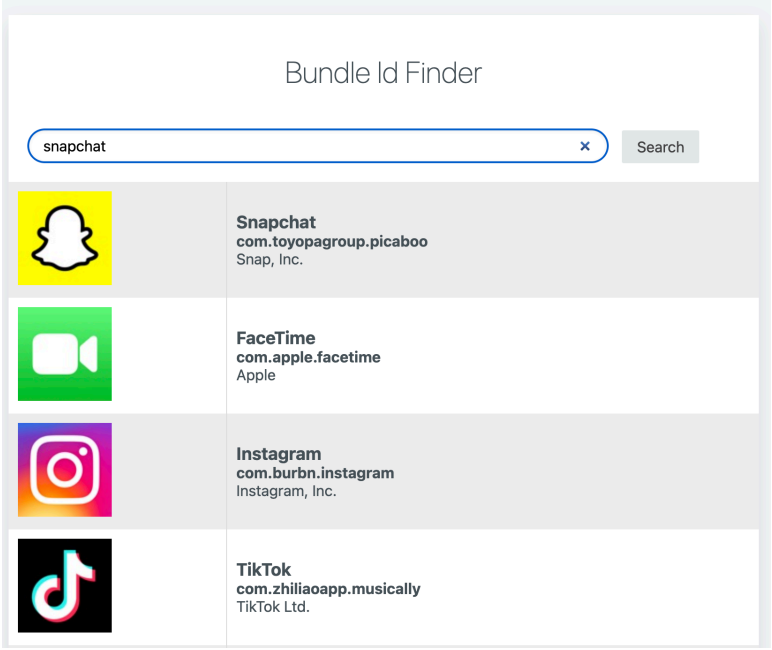
Bundle ID	Type	Directory GUID	Path
com.apple.DiagnosticExtensions.Contacts	PluginKitPlugin	001AB3DB-BD2E-4CEE-8DF6-EDA19D400CC6	/Users/abrignoni/Documents/Output/iLEAPP_Reports_2022-04-10_Sunday_164413/temp/private/var/mobile/Containers/Data/PluginKitPlugin
com.apple.PreviewLegacySignaturesConversion	AppGroup	0046CF15-671D-413A-BA5E-E9019B2365B1	/Users/abrignoni/Documents/Output/iLEAPP_Reports_2022-04-10_Sunday_164413/temp/private/var/mobile/Containers/Shared/AppGroup
com.apple.reminders.spotlightindexextension	PluginKitPlugin	004EC4B4-1597-4A25-A328-7FEC824727F8	/Users/abrignoni/Documents/Output/iLEAPP_Reports_2022-04-10_Sunday_164413/temp/private/var/mobile/Containers/Data/PluginKitPlugin

2.7 -

As discussed previously bundle IDs might not be anywhere close to their commercial or well known app names. The following URL has a useful database of iOS bundle IDs and corresponding app names:

https://offcornerdev.com/bundleid.html

See the following output for Snapchat.



2.8 -

Common Data Stores in Mobile Forensics

We have identified the locations where user generated data, by app, can be stored both in Android and iOS. Now we look at what file structures are involved. Let's start with the current king of mobile data storage.

SQLite Relational Databases

SQLite is the most used database engine in the world. It is a relation database. A simple way of thinking about them is by imagining a

set of spreadsheets that have things in common. For this example we will use DB Browser for SQLite as our tool to access SQLite databases. It can be downloaded here: https://sqlitebrowser.org/[14]

SQLite databases usually have the .sqlite or .db extension but that might not always be the case. What is always the case is that if you open a suspected SQLite database with a hex or text editor you will see 'SQLite format 3' at the start of the file. This is known as the file header and it is a sure fire way of confirming you are dealing with a SQLite database.

Using DB Browser for SQLite we will open a database named test.db for analysis. It is a simple database consisting of two spredsheets of data, tables in SQLite parlance. By using the Browse Data the content of these tables can be seen.

Customers Table

[14] https://sqlitebrowser.org/

	CustomerID	Name	Lastname
	Filter	Filter	Filter
1	1	Alexis	Brignoni
2	2	Juan	DelPueblo
3	3	John	Doe
4	4	Ellen	Chufe
5	5	Allan	Brito
6	6	Crystal	Ball
7	7	Ima	Hogg
8	8	Anita	Room

2.9 -

Addresses Table

	AddressID	Address
	Filter	Filter
1	1	480 S Keller Rd,Orlando,FL 32810
2	1	123 ABC Street, Small Town,USA 00007
3	2	8957 Lincoln Street Oakland,CA 94603
4	3	144 Glendale St.Lincoln Park,MI 48146
5	4	67 Lees Creek Rd.Maryville,TN 37803
6	5	9412 Kirkland Street Buckeye,AZ 85326
7	6	9412 Kirkland Street Buckeye,AZ 85326
8	7	101 Roberts St.Muncie,IN 47302
9	8	94 Mill Pond Street...
10	9	22 Westminster Lane Battle Creek,MI 49015

2.10 -

These tables record the customer's names and addresses. A customer can have one or more addresses in the addresses table. How to we know what addresses correspond to what customer? Notice how the customerID column in the Customers table identifies each customer uniquely. This is called a Primary Key. These same values can be found in the Addresses table under AddressID. When that is the case they are know as a Foreign Keys. The purpose of a foreign key is to identify that row of data as being part of (relational) to the primary key. We can match the customer with the correct address by finding tprimary key in the Customers table and match it with the same foreign key in the Addresses table.

To tell the database to match customers to addresses we use a set of commmands called Structured Query Language (SQL). We will tell the database to gives us all columns from both tables where the CustomerID in the customer's table is the same as the AddressID in the addresses table.

```
SELECT * FROM Customers, Addresses where CustomerID =
AddressID
```

```
1  SELECT *
2  FROM Customers, Addresses
3  where CustomerID = AddressID
```

	CustomerID	Name	Lastname	AddressID	Address
1	1	Alexis	Brignoni	1	480 S Keller Rd, Orlando, FL 32810
2	1	Alexis	Brignoni	1	123 ABC Street, Small Town, USA 00007
3	2	Juan	DelPueblo	2	8957 Lincoln Street Oakland, CA 94603
4	3	John	Doe	3	144 Glendale St. Lincoln Park, MI 48146
5	4	Ellen	Chufe	4	67 Lees Creek Rd. Maryville, TN 37803
6	5	Allan	Brito	5	9412 Kirkland Street Buckeye, AZ 85326
7	6	Crystal	Ball	6	9412 Kirkland Street Buckeye, AZ 85326
8	7	Ima	Hogg	7	101 Roberts St. Muncie, IN 47302
9	8	Anita	Room	8	94 Mill Pond Street...

2.11 -

Each customer has been match with the proper address or addresses. Notice the asterisk after the SELECT command, it means we want all columns that are responsive to the query. SQL allows us to really narrow down how much data we want from the database. If I want obtain only the addresses that are related to Alexis Brignoni we would query the database the followingt way:

```
SELECT * FROM Addresses INNER JOIN Customers ON CustomerID
= AddressID WHERE CustomerID = 1
```

```
1  SELECT *
2  FROM Addresses
3  INNER JOIN Customers
4  ON CustomerID = AddressID
5  WHERE CustomerID = 1
6
```

	AddressID	Address	CustomerID	Name	Lastname
1	1	480 S Keller Rd,Orlando,FL 32810	1	Alexis	Brignoni
2	1	123 ABC Street, Small Town,USA 00007	1	Alexis	Brignoni

2.12 -

The last query uses an inner join. These are the most common way of putting together data from two or more tables. An inner join will return rows from multiple tables when a condition is met. In our example we wanted all the data and only the data for CustomerID number 1.

SQLite databases can be pretty large with many tables and a multitude of primary and foreign keys to keep track of. Thankfully there are plenty of online resources on structured query language and with a little of patience and practice anyone can be able to pull relevant data out of these databases.

As we examine these databases we have to take into account temporary files SQLite uses as it works. These are write-ahead log and roll-back journal files. These files, if available, will have the same name as the database with either a -wal or -journal extension.

2.13 -

These temporary files might contain data not found in the database and will require examination with tools that support their forensic review. An athoritative book on the subject can be found here: SQLite Forensics by Paul Anderson[15]

JSON - Java Script Object Notation

[15] https://www.amazon.com/SQLite-Forensics-Paul-Sanderson/dp/1980293074

Chapter Three - Basic Malware Analysis

By ApexPredator

Introduction

Malware has been around for as long as computers have been in common use. Any computer program that performs malicious activities is classified as malware. There are many types of malware ranging from sophisticated self-propagating worms, destructive logic bombs, ransomware, to harmless pranks. Everyone who regularly uses a computer will encounter malware at some point.

This chapter will cover the basics of analyzing malware on an infected computer. It is targeted towards beginners who are new to Digital Forensics and Incident Response (DFIR) and hobbyists. The goal of this chapter is to show someone unfamiliar with the basic concepts of malware analysis some Tactics, Techniques, and Procedures (TTPs) used to confirm that a computer is infected with malware and how to begin extracting Indicators of Compromise (IOCs). It will cover the use of basic tools. We will not cover inter-

mediate or advanced topics such as reverse engineering malware to discover its purpose or how it works in this chapter.

The chapter will start with a little about my background and how I got started in the DFIR field. It then covers some free tools to use in basic malware analysis. The chapter culminates with a walkthrough of a canned analysis on a piece of malware. The walkthrough wraps up with recommendations on where to go next to progress to intermediate or advanced malware analysis.

How I Got Here

I started off working as a Systems Administrator and a Network Engineer. Most of my time was focused on Windows Active Directory networks. Over my career I shifted more and more towards a cybersecurity focus before eventually moving to full time cybersecurity. I completed a Master of Science in Cybersecurity and Information Assurance degree and a number of cybersecurity related certifications including OSCE3, OSCP, GXPN, GPEN, GWAPT, and GREM. The malware analysis portion of the job was one of the most interesting parts for me.

I had numerous instances of friends and family asking me to figure out why their computer was acting weird long before moving in to cybersecurity and receiving formal training on malware analysis. I have had other cybersecurity professionals ask why it is not a waste of time to learn to build Microsoft Office macro-based payloads when Microsoft if making it harder for users to run the malicious code inside to which I always respond with "Never under estimate the users desire and ability to download and run anything sent to them." People are going to download and execute malware at some point and if you are the IT expert then you will be asked to figure out what happened.

One of my first instances of basic malware analysis was when I was

in a situation that required using a computer shared by multiple people to access the internet. I erred on the paranoid side before using it to access any of my personal accounts and ran a network packet capture using Microsoft's NetMon, which is a packet capture tool similar to WireShark. I noticed from the packet capture that the machine was communicating with a Chinese domain which appeared unusual. I then conducted a quick google search on the domain and found that it was associated with a piece of malware.

The site I found listed out additional IOCs which enabled me to check running processes to see that I had the malicious executable running. I was then able to kill the process with task manager. I was also able to review the registry with regedit and delete the registry key that was created by the malware to establish persistence. I was then able to notify the other users of the machine that it had malware running on it that steals information such as account credentials. The machine was then reimaged to ensure all of the malware was removed and the machine was back to a known good state. Next, we will cover some of the basic tools that you can use to perform the same type of simple analysis.

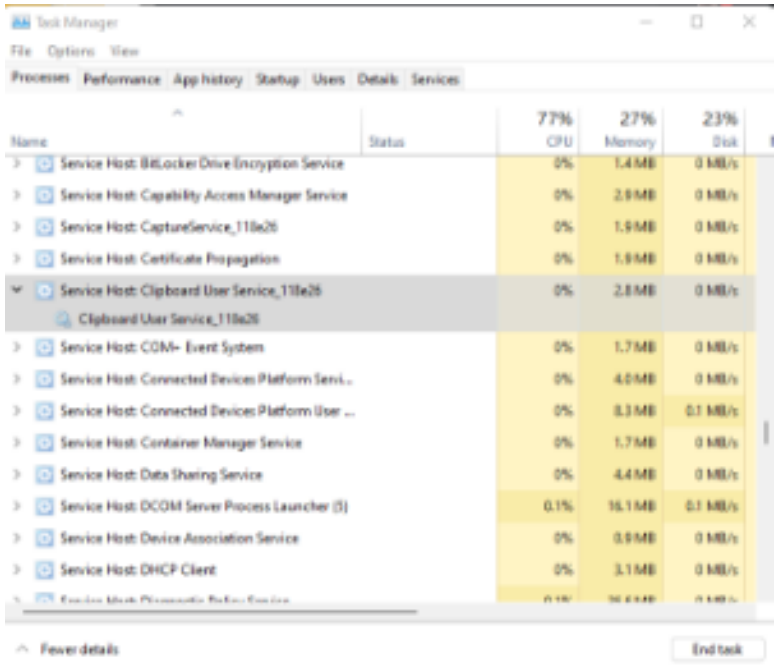
Basic Malware Analysis Tools

This section covers free tools that can be used for basic malware analysis to identify if a machine has been infected with malware. You can use these tools to extract IOCs to share with the community or to include in an Incident Response report in a professional setting. We will start with built in tools that you probably already know and discuss how to use them for basic malware analysis.

Task Manager is a built in Windows tool that allows you to view running processes. You can use it to view running processes and how much resources they are using. On Windows 10 right click the task bar and select Task Manager from the menu to launch the Task

Manager. On Windows 11 click the Windows Start Menu icon and type Task Manager to search for the Task Manager app.

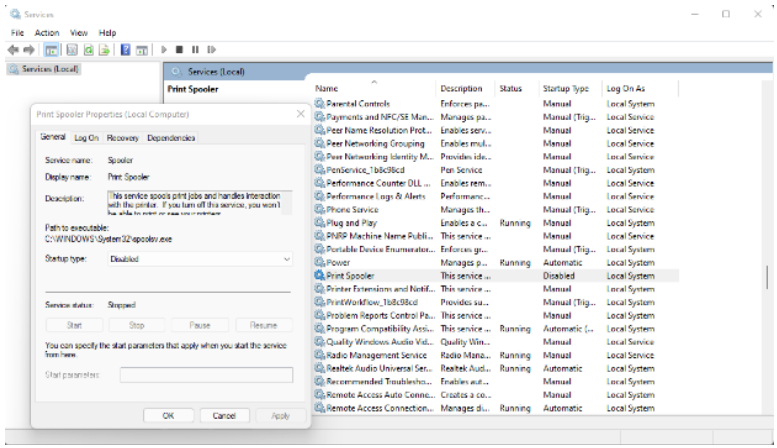
You can use this tool to look for suspicious processes running on the machine. More sophisticated Malware will attempt to blend in by using the names of common legitimate programs, however, if you have a specific process name from a IOC you can easily look to see if it is running. Each process also has an arrow you can click to expand to show child processes.



There are also ‘Startup’ and ‘Services’ tabs that allow you to view processes that are set to run on startup and the list of installed services. You can review the Startup tabs to help identify simple persistence mechanism of malware to see applications that run on startup that are uncommon or should not be included. This same process can be done on the Services tab to find suspicious services

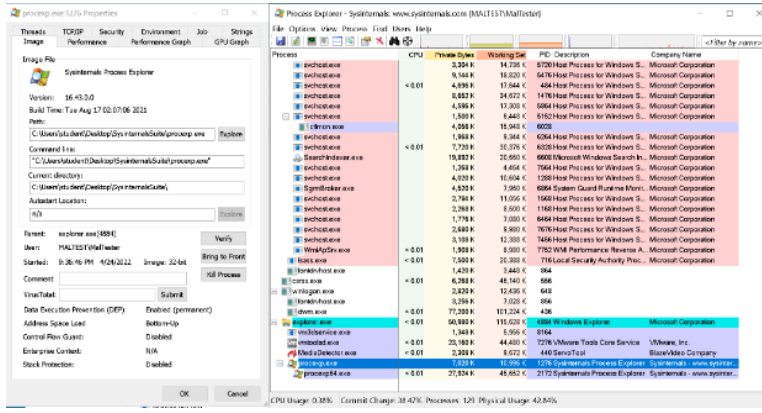
installed on the machine. These tabs show you the same information that you would get by running Startup Apps or services.msc independently from Task Manager.

You can pull up the details for each service listed in the services tab or from services.msc. It will show you the Startup type which is either manual, automatic, or disabled. The automatic startup type services will start automatically when the computer boots up Windows. You can also view the path to the executable that the service runs and what user or context it runs under. These details are useful IOCs for malicious services installed by malware.

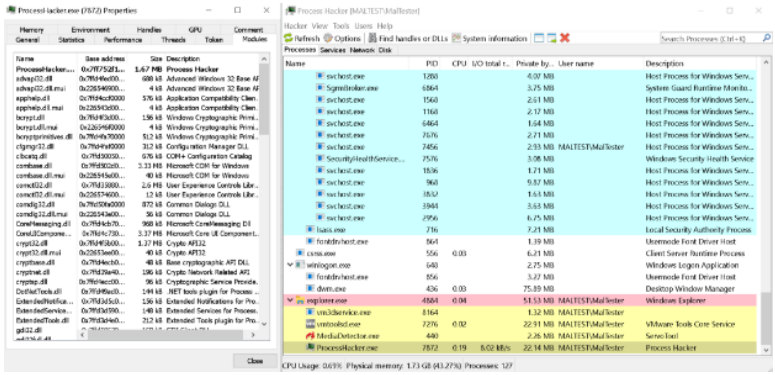


Process Explorer (procexp.exe and procexp64.exe) from the Sysinternals Suite is another free tool that provides a greater level of detail than the built in Task Manager in Windows. It provides the same functionality to kill processes while showing additional details in the main window. You can submit hashes to Virus Total thru Process Explorer to help determine if a process is malicious. It also has a tab to view TCP/IP connections showing listening addresses and ports or outbound communications made by the process. The strings tab is another great feature that allows you to see the strings embedded in the binary just like the strings command in Linux. This is useful for finding IOCs and deter-

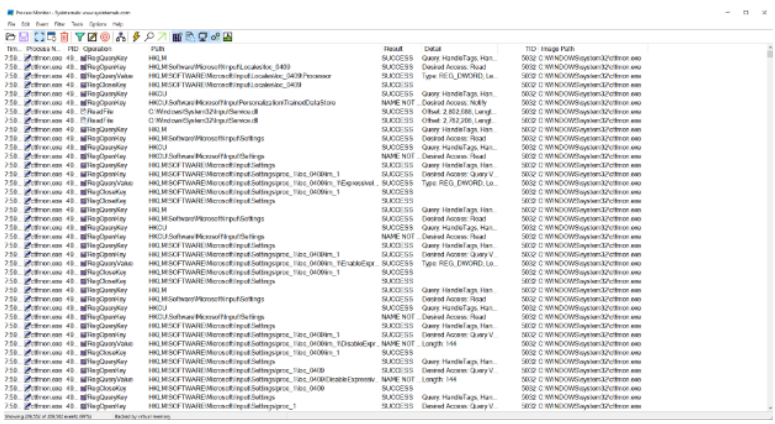
mining some of the capabilities of the malware. The Sysinternals Suite can be downloaded from https://docs.microsoft.com/en-us/sysinternals/downloads/sysinternals-suite.



Process Hacker is another great tool that performs similar functions to Task Manager and Process Explorer. It will show you the same level or process details as Process Explorer and group the processes in a parent/child process layout like Process Explorer. Process Hacker includes a modules tab when right clicking and selecting properties on a process. This modules tab shows you all of the modules loaded and in use by the process. This is helpful for finding additional IOCs or identifying malicious dll files used by a suspicious process. Process Hacker is available for download at https://processhacker.sourceforge.io/downloads.php.

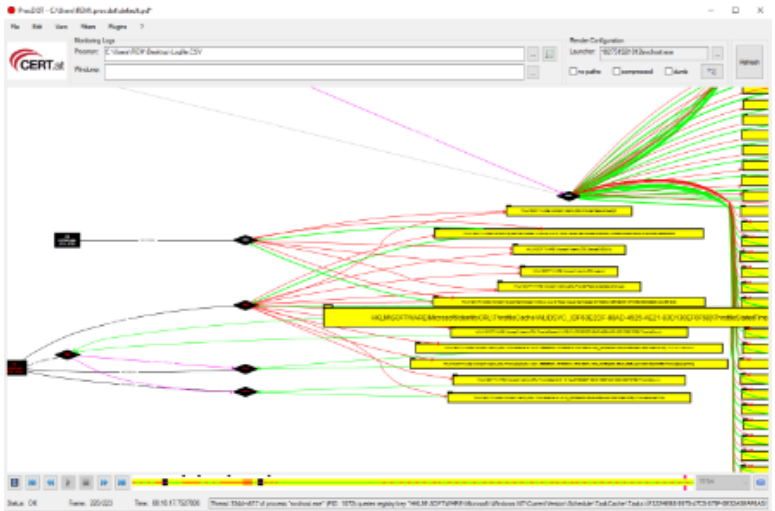


Process Monitor, or Procmon, is another tool included in the Sysinternals Suite that is usefull for monitoring processes. Procmon goes beyond the process information provided by Task Manager, Process Explorer, or Process Hacker. It shows every action taken by the process allowing n-depth analysis of suspicious or malicious processes. Procmon will quickly overload an analyst with data unless filters are used to filter out the noise. It enables an analyst to find IOCs and understand what actions the malware has taken on the system.

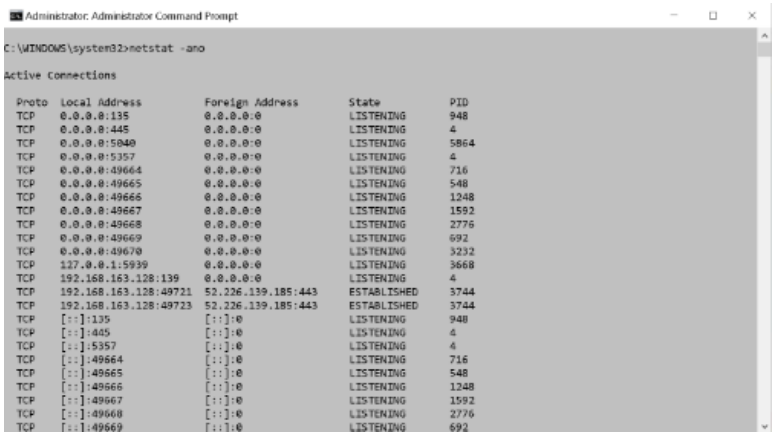


ProcDOT is useful for filtering and displaying the results from Procmon. ProcDOT allows an analyst to ingest the logs generated

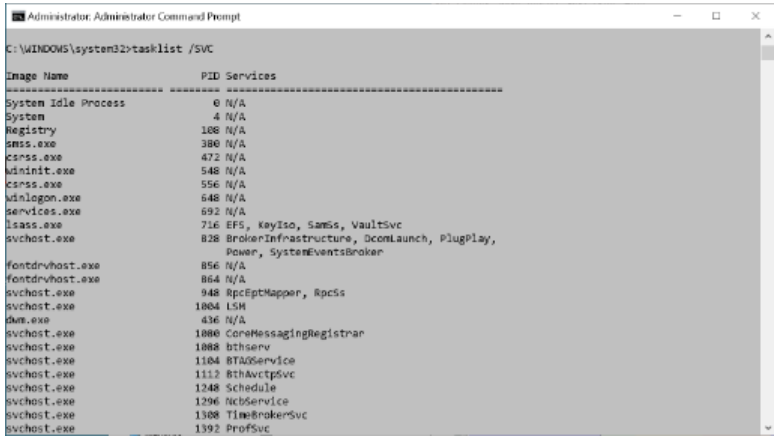
from a Procmon capture saved in a CSV file. The analyst can then select the desired process from the imported CSV file and ProcDOT will generate an interactive graph. This effectively filters out the noise of unrelated processes giving the analyst an easy-to-follow graph that shows all actions conducted by the malware to include those of child processes spawned by the original process. It also allows to ingest packet captures to corollate with Procmon. ProcDOT can be downloaded from https://www.procdot.com/downloadprocdotbinaries.htm.



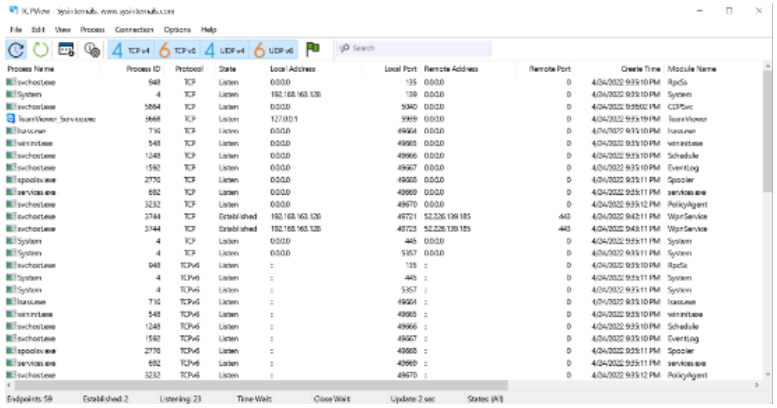
The netstat tool included in Windows is another useful tool. You can use it to view all listening ports and established connections. You can view the connections and listening ports with the command *netstat -ano*. This command includes the process ID of the process using that listed port to help you correlate a suspicious connection to a process.



The tasklist command can be used to list running process and their associated process ID from the command line. This can help you enumerate suspicious processes with out needing to use a Graphical User Interface (GUI). It is helpful when used in conjunction with netstat to look up the process ID found with a suspicious network connection. The below screenshot shows that PID 4 listening on port 445 (RPCSMB) on all interfaces (0.0.0.0) is the System process. In this case it is a legitimate process and listening port combination. The System process also always loads at PID for so if it were a PID other than 4 that would be unusual and a potential IOC.



Another way to do the same analysis is to use the TCPView tool from Sysinternals Suite. The TCPView tool provides the same information received from *netstat -ano* and *tasklist /SVC* in a convenient and easy to read GUI. This allows you to quickly identify suspicious listening ports or connections and correlate them to the corresponding process. The remote address listed in TCPView and netstat is another useful IOC to include in your analysis.

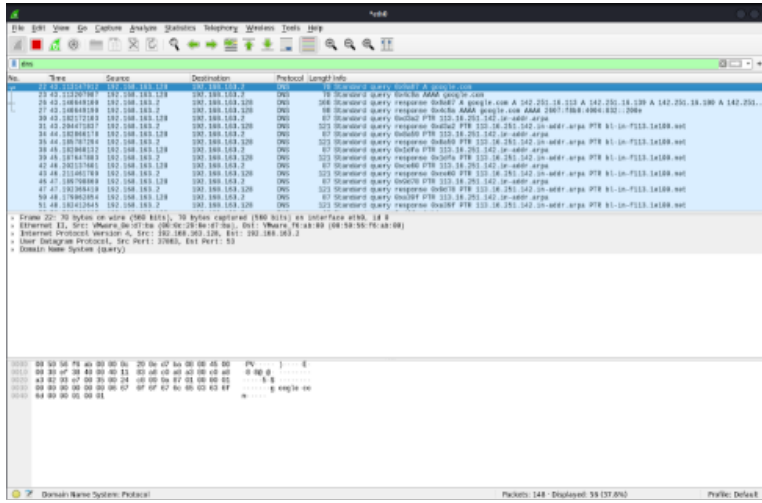


Wireshark is a valuable tool to conduct more in-depth packet

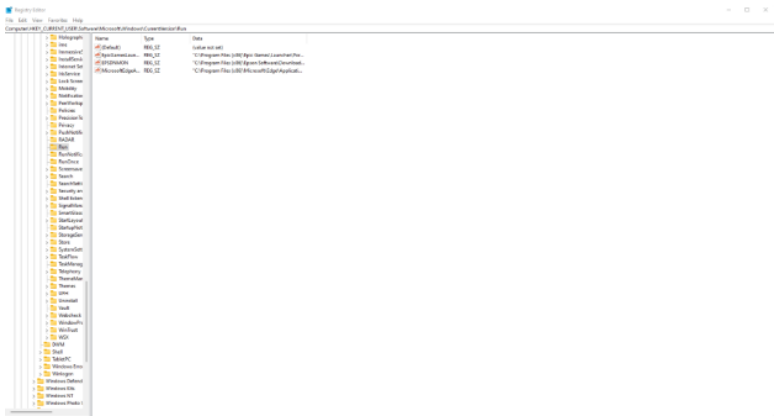
analysis. Wireshark enables a malware analyst to view all network traffic sent and received on the suspected machine. An analyst can filter the packets by IP, port, protocol, or many other options. Filtering by DNS protocol enables an analyst to find DNS queries to malicious sites used for Command and Control (C2) of malware. The domains found in the DNS queries are useful IOCs to determine if the machine is compromised.

Wireshark provides capabilities to conduct more advanced analysis of malware communication. It allows an analyst to identify C2 traffic hidden in protocols such as DNS. It also enables an analyst to extract data such as second stage binaries or infected text documents downloaded by the malware. Using a proxy in combination with Wireshark enables an analyst to export the certificate and keys used to encrypt Transport Layer Security (TLS) encrypted traffic to recover the plaintext data sent between malware and attacker-controlled servers.

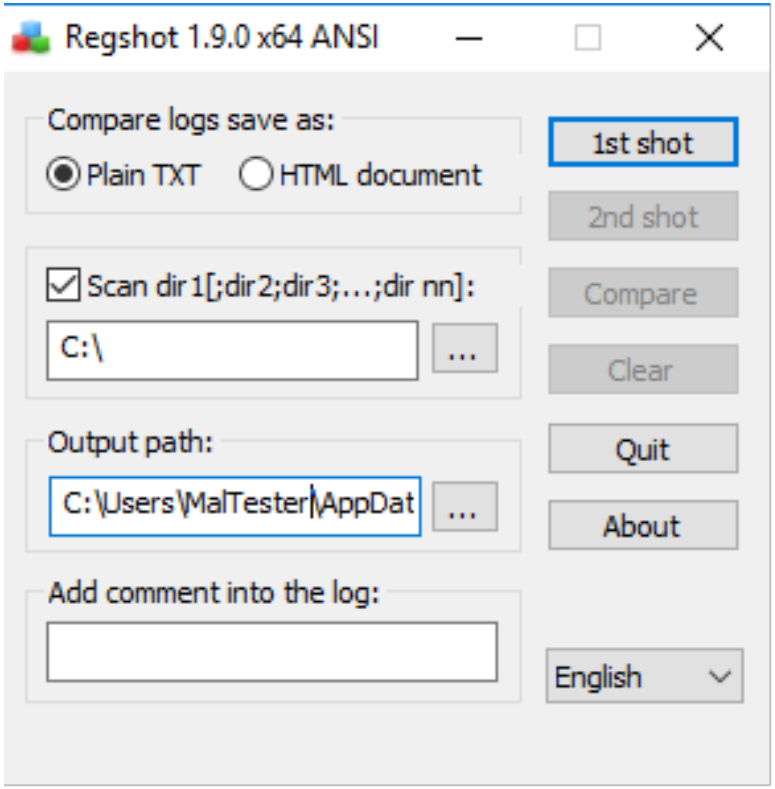
The malware analysis walkthrough in this chapter will focus on using Wireshark to perform basic analysis tasks. This includes reviewing DNS queries to identify suspicious domain look ups and plaintext commands/passwords sent during malware communication. More advanced usage of Wireshark is out of scope of basic malware analysis and is saved for future writings on intermediate and advanced malware analysis. Wireshark can be downloaded from https://www.wireshark.org/. Microsoft's NetMon is an alternative to Wireshark, but is only available for download from archive and is no longer being developed https://www.microsoft.com/en-us/download/4865.



Regedit is another useful tool built in to Windows. Regedit gives the ability to view and edit the Windows registry. It can be used for basic malware analysis to search for persistence mechanism such as entries in HKEY_LOCAL_MACHINESOFTWAREMicrosoft-WindowsCurrentVersionRun or HKEY_CURRENT_USERSoft-wareMicrosoftWindowsCurrentVersionRun. Applications listed in the run keys will auto start when a user logs in to the machine and is sometimes used by malware to establish persistence.



Regshot is useful for determining what changes an application makes to the Windows registry when it is executed. Regshot allows an analyst to take a snapshot of the Windows registry before and after executing a suspicious application and generates a comparison of the two snapshots. This is useful when analyzing a suspicious application in a controlled lab setting. Regshot can be downloaded from https://github.com/Seabreg/Regshot.



Certutil is another tool built in to Windows that is useful for malware analysis. An analyst can use certutil to generate a hash of a file to compare it to a known malicious file hash. This can indicate if a file is malicious without having to execute it to investigate what it does. An analyst can use the hashes generated by cerutil as IOCs once a suspicious file is determined to be malicious thru analysis.

An analyst can also use automated tools for analysis. A suspicious file can be uploaded to Virus Total (https://www.virustotal.com/gui/home/upload). Virus Total is an online system that will execute the file in a sandbox to attempt to determine if it is malicious or not. It will then provide file hashes and IOCs an analyst can use to identify the file. Virus Total also shares uploaded files with

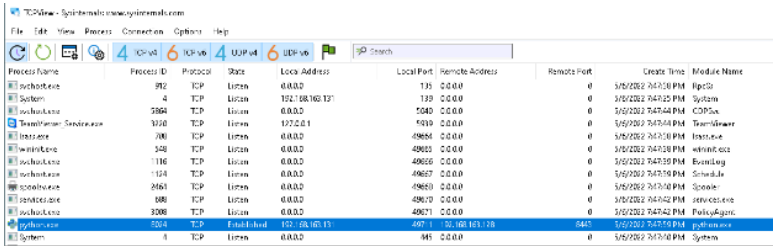
Antivirus vendors to use for building detection signature.

Basic Malware Analysis Walkthrough

It is time to do a walkthrough of a sample malware analysis now that you are familiar with some of the tools used for malware analysis and their capabilities. The walkthrough will show how to use some of the tools mentioned in this chapter. It will not use any tools not previously mentioned.

In this scenario a user has reported to you that their machine has been running slow and acting "weird." You have already conducted initial investigations by asking the user questions including: "When did the issues start?" "Did you download or install any new applications?" "Did you click any links or open any documents from untrusted sources?" The user states that they did not install any application recently but did review a Microsoft Word document sent from a customer.

We start our analysis with opening TCPView from the Sysinternals Suite to see if we can quickly spot any unusual processes communicating to remote sites. In this simple scenario we see that there is currently only one process, python.exe, communicating to a remote system. We flag this as suspicious since python is not typically used in the manner for our fictious network. We then make a note of the port and IP for potential IOCs.



We then open up Wireshark to review a packet capture of the incident. We use the IP and port combination to filter the traffic down to the currently interesting packets. The traffic is not encrypted which will allow to view plaintext communications. We then Right click on one of the packets and select followTCP stream to pull up the conversation in a readable format. This shows that this is a malicious process used to create a reverse shell to the attacker. We are able to identify commands sent by the attacker and the response from the infected machine.

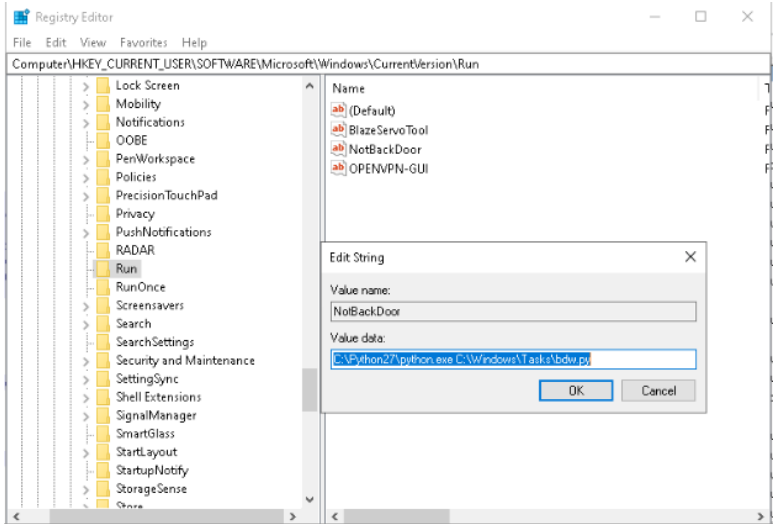
The attacked ran a series of command to enumerate identifying information about the machine and what privileges the user account has. The attacker then attempts to establish persistence by creating a service named “NotBackDoor” to auto start the malware containing the reverse shell. This action failed leading the attacker to then attempt persistence thru creating a run key in the system

registry for the current user and was successful.

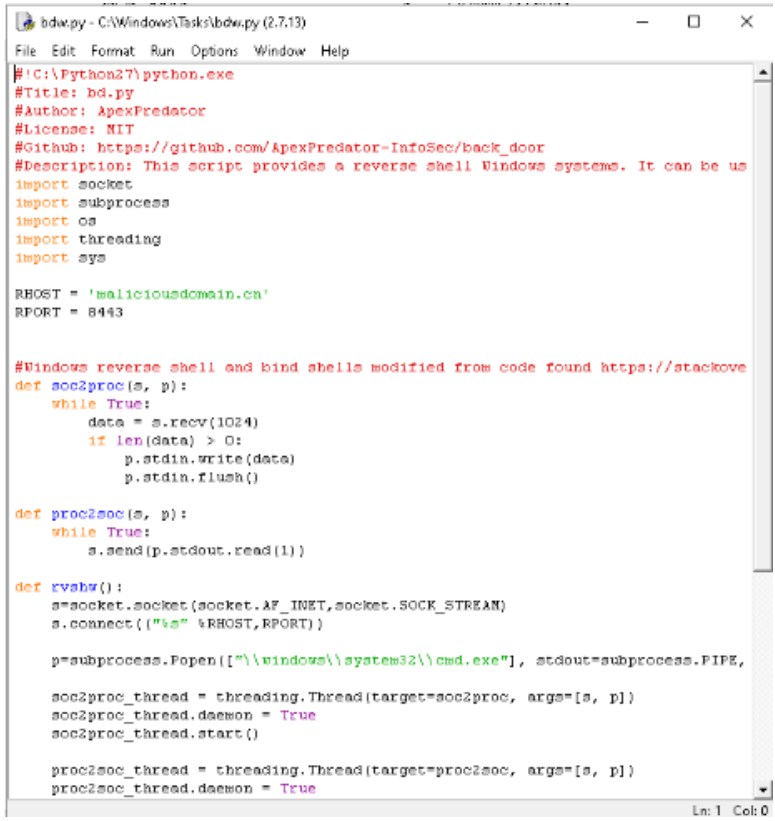
At this point we have verified that there is malware present on the system and it is actively being exploited by a remote attacker. We immediately take action to isolate the machine to cut off access from the attacker and protect the rest of the network. In this scenario we would just simply block the IP and port on the perimeter firewall and remove the infected machine from the network before continuing our analysis.

We then take steps to confirm the persistence measures taken by the attacker. We review the services in services.msc to verify that NotBackDoor service was not successfully created. Then we take a look to ensure no other unusual service exist. The NotBackDoor service name and the binPath option of "C:Python27\python.exe C:WindowsTasks\bdw.py" are still noted as IOCs since the attacker did attempt to create the service and it could be present on other infected machines if access was granted.

Regedit is then used to verify the run key created after verifying that no malicious services exist. We do find a NotBackDoor key that points to "C:Python27\python.exe C:WindowsTasks\bdw.py". We make not of this as an IOC. We also note that C:WindowsTasks\ is commonly used as a location to drop malware due to low privilege users being able to write to the location and is common excluded from protections such as application whitelisting since it is located under C:Windows.



The next step for this scenario is to navigate to the C:WindowsTasks\ folder to investigate the bdw.py file mentioned in the previous steps. The investigation shows that this is just a simple python script to establish a reverse shell from the infected computer to the attacker's machine. We are able to see that it contains the port number 8443 but is pointing to a domain name of "maliciousdomain.cn" instead of IP. We add this domain to the list of IOCs. We could have also identified the traffic associated with this domain if we had started this investigation by looking for suspicious DNS calls. The .cn root domain indicates this is a Chinese domain and if we are in a scenario where traffic to China is abnormal then this is a potential red flag.

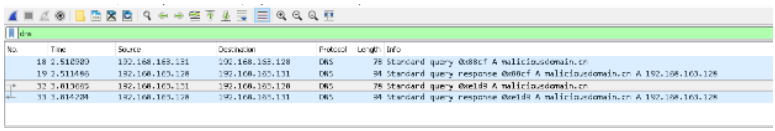


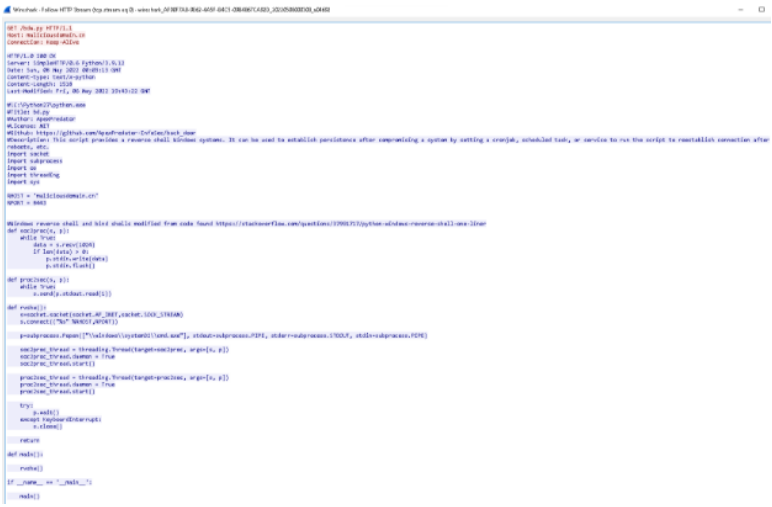
We know that bdw.py is malicious and provided a remote attacker access to the infected machine, but we do not yet know how it go there. We see that the document the user stated they received from a new customer ends with the extension .docm. This informs us that the document contains macros which could be the initial infection vector. Analysis on this file needs to be done in an isolated lab to prevent any reinfection.

The document in this scenario contains only one line of text stating that it is a generic document for a malware analysis walkthrough. We could look for unique strings in the document that could be used for IOCs in a real-world scenario to help others determine if they have received the same document. The next step is to check the documents for macros.

Click view in the ribbon menu at the top of the document. Then select the Macros button and click the edit button in the window that pops up. We can see that this document does contain a simple macro that uses PowerShell to download bdw.py from malicious-domain.cn. The macro then executes bdw.py to initiate the initial reverse shell connection. The macro contains the AutoOpen and Document_Open sub routines to run the downloader when the document is opened. We have now verified that Doc1.docm is a downloader used to infect the system with a python based reverse shell. We add Doc1.docm to our list of IOCs.

We could have started our analysis with the Doc1.docm document that was mentioned by the user. This would have given us the info to track down the reverse shell that we had found by analyzing the network traffic and processes earlier. Running Wireshark while executing the macro shows us the DNS calls to the maliciousdomain.cn. We can also extract the bdw.py from the HTTP stream since it was download unencrypted via HTTP. This can be useful in instances were more advanced malware downloads another stager and then deletes the stager from the system after running its payload.





We can also use the built in certutil.exe tool to generate hashes for the malware files to use for IOCs. Run 'certutil -hashfile Dco1.docm SHA256' to generate a SHA256 hash of the document. You can also generate and md5 hash and generate the hashes for the bdw.py. These are useful IOCs for signature-based systems to detect the presence of the malware.

Administrator: Command Prompt

```
C:\>certutil -hashfile C:\Doc1.docm SHA256
SHA256 hash of C:\Doc1.docm:
6fa2281fb38be1ccf006ade3bf210772821159193e38c940af4cf54fa5aaae78
CertUtil: -hashfile command completed successfully.

C:\>certutil -hashfile C:\Doc1.docm md5
MD5 hash of C:\Doc1.docm:
b85e666497ea8e8a44b87bda924c254e
CertUtil: -hashfile command completed successfully.

C:\>certutil -hashfile C:\Windows\Tasks\bdw.py SHA256
SHA256 hash of C:\Windows\Tasks\bdw.py:
f24721812d8ad3b72bd24792875a527740e0261c67c03fe3481be642f8a4f980
CertUtil: -hashfile command completed successfully.

C:\>certutil -hashfile C:\Windows\Tasks\bdw.py md5
MD5 hash of C:\Windows\Tasks\bdw.py:
34ca38da117d1bb4384141e44f5502de
CertUtil: -hashfile command completed successfully.

C:\>
```

We can use Procmon and ProcDOT to verify that the malicious files did not spawn any additional processes that need to be investigated. The ProcDOT graph shows us that the python.exe process communicated over TCP to IP 192.168.163.128 and spawned a cmd.exe process. We can see the commands that were run in the cmd.exe in the graph and verify that no additional files or processes were created.

We have now completed analyzing the system to verify that it is infected with malware. We determined what the malware does and we have extracted IOCs to implement in our defensive tools to detect future infection attempts. The machine will need to be reimaged before returning it to the user for use to ensure all malware has been eradicated. It is important to ensure a forensic image is taken before reimaging the system if evidence preservation

is needed to legal cases or future investigations. To recap our IOCs:

- Downloader macro in document title "Doc1.docm"
- Unique string "This is a generic document for a malware analysis walkthrough" in Doc1.docm
- Second stage python reverse shell named bdw.py stored in C:WindowsTasks\
- Service named NotBackDoor to auto start bdw.py
- HKCU\ SOFTWAREMicrosoftWindowsCurrentVersionRun-NotBackDoor registry key to autorun bdw.py
- SHA256 hash of Doc1.docm 6fa2281fb38be1ccf006ade3bf210772821159193e38c9
- Md5 hash of Doc1.docm b85e666497ea8e8a44b87bda924c254e
- SHA256 hash of bdw.py f24721812d8ad3b72bd24792875a527740e0261c67c03fe34
- Md5 hash of bdw.py 34ca38da117d1bb4384141e44f5502de
- Bdw.py downloaded from maliciousdomain.cn
- Bdw.py reverse shell to IP 192.168.163.128 (maliciousdomain.cn)
- Bdw.py reverse shell on port 8443

This was a simple example of how to conduct basic malware analysis. The tools ant technique discussed in this scenario can be used in a real-world scenario to determine if a machine is infected with malware and extract some IOCs. The malicious files used for this scenario and a copy of the walkthrough can be found on my GitHub (https://github.com/ApexPredator-InfoSec/Basic-Malware-Analysis). You will need a system with netcat to receive the reverse shell as well as fakedns (https://github.com/SocialExploits/fakedns/blob/main/fakedns.py) to simulate a DNS server to direct the malicousdomain.cn calls to your attacker machine.

More advanced malware will require additional tools and techniques. The techniques to reverse engineer malware to include decompiling, disassembling, and debugging is covered in course such as the SANS FOR610 Reverse Engineering

Malware (https://www.sans.org/cyber-security-courses/reverse-engineering-malware-malware-analysis-tools-techniques/). The FOR610 course is a good step up to the next level if you enjoyed this basic malware analysis. The course also teaches some techniques for deobfuscating code where as this basic analysis only consisted of unobfuscated code.

Additional advanced topics to look into include techniques to recover encryption keys. Those techniques are useful to unencrypt source code of encrypted malware or to help recover keys to unencrypt files that have been encrypted by ransomware. Assembly language programming familiarity is needed for debugging and reverse engineering of malware. Basic knowledge of JavaScript is also useful for analyzing web-based malware.

You can also increase your skills by taking malware development course from Sektor7 (https://institute.sektor7.net/red-team-operator-malware-development-essentials). Learning to develop malware will help you better understand how to detect malware and will teach you additional techniques used by modern malware. SANS also offers the advanced FOR710 course for advanced reverse engineering malware (https://www.sans.org/cyber-security-courses/reverse-engineering-malware-advanced-code-analysis/).

If you enjoyed this walkthrough would like to see more you can check out my Github for a walkthrough on performing white box code analysis of a vulnerable web application and coding a full chain exploit. I have solutions for various vulnerable web applications and binary exploitation challenges and will be adding a couple of binary exploitation and reverse engineering walkthroughs in the future. I can also add in intermediate malware analysis walkthroughs if there is enough interest.

Chapter 4 - I Have a Password Hash, Now What?

By John Haynes[16]

Disclaimer / Overview

This chapter is a beginner's guide on how to crack passwords. While on the surface this may seem to be something reserved for cybercriminals, there are legitimate reasons for a law-abiding individual to understand this process. Firstly, those who work in penetration testing or a red team environment will need to know how to do this task. Secondly, law enforcement may need to access data that is password protected with the legal authority of a search warrant. Third, important data may need to be recovered from a device after the owner is deceased for the estate or heirs. There may also be other ways to legally access password protected data such as forgotten passwords or security concerns in a courporate

[16] https://www.youtube.com/channel/UCJVXolxwB4x3EsBAzSACCTg

envornment. Finally, it is important for someone who wishes to keep their data secure to understand this process to know why a strong password is important and how to test the security of their passwords without compromising those passwords.

That being said, I do not condone, encourage, or support those who would use this information for malicious or illegal means. This paper will start with the fundamentals of hashing and end with showing how a strong password makes a substantial difference when attempting to crack complex passwords.

In digital forensics, the first challenge is to actually get the data in a state so that it can be analyzed. For those that need to legally access the data, there should be something in here for you. For those that wish to learn how to better secure their own data, there should be something in here for you as well. Let's get started!

Password Hashes

At the fundamental level, a password is like a key that fits into and unlocks a particular lock. Only you have the key, but anyone can come up and inspect the lock. With a mechanical lock, nobody can see the internal functions of the lock without specialized tools like lock picks. If someone was proficient at using lockpicks, they could theoretically determine the depth of each pin while picking the lock to make a key that would unlock the lock.

The same sort of concept is true for passwords. Each password should have a unique algorithmic hash. To obtain a hash, a complex mathematical algorithm is run against a string of data and the output is an extremely unique character string. For some weaker hash algorithms, there have been hash collisions where two different sets of data have resulted in the same outputted hash. However, when considering human-generated passwords, it is normally not necessary to worry about hash collisions. It is sufficient to say that

if you have the hash of a password you have the password in an encrypted state. The password hash is how the password is stored on any modern operating system like Windows, macOS, or Linux or for encrypted containers like Bitlocker or encrypted 7zip files. With the right tools, that is the only part of the password that will be available for an examiner to inspect, just like the external part of a lock is the only thing to inspect on a locked door if someone were to try and pick the lock. There are methods to prevent the extraction of a password hash, but it is reasonable to attempt to find a method to extract a hash from a system if the individual has physical access to the electronic device, encrypted file, or a forensic image (.E01, dd, or similar) of an encrypted volume.

Therefore, if the password hash can be extracted, it can be attacked to attempt to crack the password. Hashing algorithms are mathematically a one-way operation. If someone has a password hash, there is no normal mathematical operation that can be performed to reverse engineer the original plaintext password. Additionally, some hashing algorithms are more difficult to crack than others because the speed of decryption is sacrificed for security. However, the user can guess the potential password, hash it, and then compare the resulting hash against the known hash. If it is a match, then the password is cracked. This would be a very slow method to do manually, but there is software like Hashcat that can be used to automate this process to perform thousands of attempts per second. To make the guessing more difficult, the system can implement what is known as "salt" into the hash to obfuscate the hash and make it more difficult to crack.

A discussion of password hashes would not be complete without mentioning salted passwords. The salt for a password is additional data that is added to the password before the hash algorithm is applied to complicate the guessing of the password. Therefore, the salt would have to be known and applied to each potential guess otherwise the hash would be incorrect even if the correct password was guessed. The salt can be generated in several different ways

and can be static or dynamic depending on developer choice. Unfortunately, Windows does not salt the NTLM password hashes that it generates so they are vulnerable to attack.

As was just mentioned, Windows stores password hashes in NTLM format. This is unfortunately a very weak form of encryption as it is the equivalent of MD4 encryption. The VTech company was compromised in 2015 by a SQL injection attack and when the password hashes were analyzed they were determined to be encrypted with MD5. MD5 is considered to be a weak form of encryption and some do not consider it to even be encryption as it is so weak. Windows uses even weaker encryption for its passwords, and those passwords are not even salted to compensate for the weak encryption! Windows has upgraded to NTLMv1 and NTLMv2 for some uses, but those are still weak by most encryption standards. Even more concerning is these NTLM hashes of user passwords are transmitted over the network for authentication between computers (Patton, 2022). This is one of the most common passwords that users will use and can be extracted by several methods, including packet sniffing. It is also nearly guaranteed to not be generated by a password manager as the user has to physically enter the password into the keyboard.

Useful Software Tools

There is no reason to reinvent the wheel and in most situations, someone else has already created a tool that will perform the task needed. The general workflow for cracking a password is hash extraction, hash identification, attacking the hash with general methods, and attacking the hash with custom methods. Tools that can assist in these phases are Mimikatz, Hashcat, John the Ripper, Passware, Gov Crack, custom scripts often shared on GitHub and many more. Some tools like Passware are paid tools, and while there is nothing wrong with a paid tool this paper will focus on

using the free tool called Hashcat. As was just mentioned, Gov Crack has a graphical user interface (GUI) while Hashcat and John the Ripper use command-line interfaces (CLI). Normally GUI interfaces allow for ease of access but tend to lack the flexibility of CLI tools. Nearly all of the custom scripts that are used for hash extraction and are posted on GitHub are going to be CLI-based tools. If the reader is unfamiliar with the command line, that should not be a limiting factor for at least understanding the methods discussed in this paper and there will be step-by-step instructions on how to crack a password hash in Hashcat. The focus on a particular set of tools over another is due to personal experience with certain tools and no bias towards any particular tool is intended as many tools can do the same thing and overlap with each other with certain functions.

Hash Extraction Techniques

One common method to extract an NTLM hash is to use Mimikatz, but it is widely recognized as malware by most anti-virus software. Due to the volatile nature of Mimikatz, I will not be providing information on where to obtain it as I obtained my copy in person from a trusted source. However, if the individual has access to the forensic image (an .E01 or similar) of the hard drive of the computer, then Mimikatz should be used against the SAM and SYSTEM registry files found in C:WindowsSystem32Config, assuming Bitlocker or another form of encryption is not present. Even with live access to a machine, administrator rights and a forensic tool such as FTK Imager, preferably preloaded on a USB drive, will be required to copy the registry files as a simple copy/paste or drag-and-drop method will not work. This is just one way to obtain an NTLM hash as it can also be obtained by observing network traffic. In general, this is a great place to start when trying to crack passwords and try out different methods as the NTLM hash uses weak encryption.

If the examiner is looking at an APFS encrypted volume from a Macbook, it is important to realize that the password for the encrypted volume is the same as the password used to log into the system. However, this hash uses a strong encryption method and will take much longer to crack as compared to an NTLM hash. To extract the hash, there are tools available like the one from user Banaanhangwagen on GitHub. https://github.com/Banaanhangwagen/apfs2hashcat. This will require using Linux to run the tool and extract the hash from a raw or .dd forensic image.

Other strong encryption methods include Bitlocker, zipped or compressed files, and many more. Generally speaking, some smart person somewhere has found out how to extract the hash and has shared that information for that particular situation. The examiner needs to search for hash extraction of a particular make, model, file system, software version, or a combination of those and similar attributes. Also as a general rule, the hash is likely to be stored in plain text somewhere in the hex (the raw data) on an electronic device. If the examiner is willing to poke around and search the hex, they may be able to find the password hash assuming they are correctly decoding the hex. This is not a hard-fast rule by any means, as there are complex methods of preventing external access to protected memory areas. For example, I know of no known method at the time of writing this to extract a hash from a Chromebook even though it is possible to log into a Chromebook without it being connected to the internet, implying that a hash of the user's password must be stored locally on the device.

Hash Identification

There may be times when a password hash has been located but the hash type is unknown. Hashcat has an entire wiki including example hashes that can aid in this process. The example hashes are

located at https://hashcat.net/wiki/doku.php?id=example_hashes and can help with the hash identification of an unknown hash. A simple Google search for "Hash Identification" results in multiple online tools that can help identify the type of hash, be it NTLM, SHA-256, or many others. Several websites include https://nth.skerritt.blog/, https://hashes.com/en/tools/hash_identifier, or https://www.onlinehashcrack.com/hash-identification.php. Be wary of using these or any other websites for sensitive hashes as the website now has the actual hash. For advanced examiners who do not want to use an online tool, Kali Linux also has an offline tool that can be downloaded and used locally.

Attacking the Hash

Once the type of hash is identified, it is time to attempt to crack the hash. The simplest yet least secure method of cracking a password from a hash is once again to use an online resource. Some of the previously mentioned websites also offer services that will attempt to crack a hash, but those are limited. The use of a password cracking tool such as Hashcat is highly recommended as it allows for a much more powerful, robust, and secure method of cracking a password hash.

Here is a hash taken from the Hashcat Wiki (b4b9b02e6f09a9bd760f388b67351e2b). This is an NTLM hash of a word in the English language. If you have visited the website then it is easy to determine what this hash is, but let's assume that we know nothing about this hash other than it was extracted from a machine and we wanted to crack this hash using Hashcat. Recall that the method of cracking this password has to be coming up with our potential password, hashing it, and comparing the two hashes until we find a match. This is a process Hashcat will automate for us. So if we get it wrong, the worst that will happen is we will move on to the next potential password and try again. Therefore, there are two primary

methods of attacking a password, a brute-force method, and a more focused attack. An exhaustive brute-force attack would take the combination of all possible symbols on the keyboard and iterate through them. This is not ideal, but let's explore the mathematical reason why it is not the best method before explaining a better method.

If an exhaustive attack was to be performed against a password, that would mean that every possible permutation of all possible characters, numbers, and symbols on the keyboard would be attempted. For the standard English QWERTY keyboard, there are 10 digits "0123456789", 26 lowercase letters "abcdefghijklmnopqrstuvwxyz", 26 upper case letters, "ABCDEFGHIJKLMNOPQRSTUVWXYZ", and 33 special characters or including symbols, "!@#$%^&*()-_-=+[|;:'",<.>/? ". Note that space or the spacebar is also included in the special character count. Adding these together results in 10 + 26 + 26 + 33 = 95 or ninety-five total possible characters that can be used at any point in a password, assuming they are all allowed for use in a password. So for a single character password, there are only 95 possible combinations. For a two-character password, there are 95 x 95 = 9,025 possible combinations. A three-character password has 95 x 95 x 95 (or 953) = 857,375 combinations, a four-character has 954 = 81,450,625 combinations, and a very short five-character password has an astonishing 955 = 7,737,809,375 password combinations, over seven billion! Even a meager eight-character combination has over six quadrillion (a quadrillion is the name of the number just beyond trillion) possible combinations for just the eight characters alone! Not only does this show the difficulty of using every possible character, but it also shows the strength of using unusual symbols in passwords. Even with modern computing that is capable of computing thousands of possible passwords per second, it could take decades or longer to attempt to crack an eight-character password using this method using normal computers. We need a better method!

So to speed up this process we need to make some assumptions

about the original password rather than guessing random characters. This brings up the primary weakness and therefore the best method of attacking passwords once the examiner has the hash. Since most passwords must be remembered by the user, it is very likely to contain a word in a language that the user knows. The total number of guesses can be greatly reduced by avoiding letter combinations that are not words. The total number of words in the 2022 Oxford English dictionary is over 600,000 words, but this does include outdated, obsolete, and obscure words. Still, this is a huge improvement over even a short three-letter permutation!

It is also common to add numbers or symbols to the end of the password. So we can also add numbers to the end of a valid word and try those combinations. Sophisticated users may decide to use "leet speak" and replace letters like 'S' with the number '5', the letter 'A' with the number '4', the letter 'E' with the number '3', the letters 'I' or 'l' with the number '1' because they look similar to the corresponding letter. For example, the word "Apples" may become "4pp135" Finally, the addition of symbols is common at the end of the password, so common symbols like "!" can be added to the end (Picolet, 2019). This is by no means an exhaustive list, but this is a good starting point considering the alternative of a true brute force attack.

Wordlists

Now that we know a better method, we need to come up with a way to use that method to attack passwords. The simplest method would be to use a list of words or a wordlist of possible passwords. Just like it sounds, it is a list of possible passwords that already have symbols and numbers added to them. When using a wordlist to attack a password, it is known as a dictionary attack. It is possible to manually build our wordlist, but that is a very time-intensive task as we would not only need to create useful passwords but avoid

duplicates. Fortunately, there are prebuilt wordlists that we can use.

When companies are hacked, a part of the data that is often stolen is the passwords. Companies should encrypt their data, specifically user passwords, but this is not always the case. In 2009, the social gaming company RockYou was compromised by a SQL injection attack. The hacker was able to gain access to over 32 million accounts and they were storing passwords in the clear, which means that there was no encryption whatsoever on the passwords as they were stored in plain text (Cubrilovic, 2009). This list of passwords has become known as the rockyou list and is commonly used as a starting point for dictionary attacks. Future breaches where the passwords have been compromised and cracked have also been added to wordlists. It is important to note that a good password list will not have duplicates of passwords, or the passwords have been deduplicated. This is a key way to save time when cracking passwords.

A good online resource where wordlists are compiled and ranked is Weakpass.com https://weakpass.com/ (W34kp455, 2014). On this site, wordlists are ranked by order of popularity and functionality from 0 to 100 and using a color-coding system that corresponds with the numerical ranking. Note how there are varying sizes of lists, ranging from over 400GB to only a few bytes in size. The first several wordlists for download may not be ranked very high being color-coded red and only being in the single digits. Selecting "Lists" and selecting "Medium" should display the original rockyou wordlist as rockyou.txt on the first page with just over 14 million unique passwords. When selecting "Lists" from the horizontal menu and selecting "All" we can sort all lists by popularity. Near the top of the list should be the "cyclone.hashesorg.hashkiller.combined.txt" password list with about 1.5 billion total passwords. This list is one of the top-ranked lists while only being just over 15GB in size. I would recommend using this list and I use it frequently due to its good combination of reduced size compared to other lists and overall speed

in password cracking. The total time to iterate through the list is not unreasonable for many password hash types and stands a decent chance of cracking the password with a straight dictionary attack. The All-in-One tab allows for downloading a deduplicated version of all passwords on the site in various lengths for different applications, but know that a longer list will take longer to complete than a shorter list. If you haven't noticed, there is also an estimated time to iterate through the list for a particular password type under each list. While this can vary widely between different computers, it does a good job of showing the relative time difference it takes to attempt that list against the different hash types. If the 15GB password list is too large for you, another smaller list that is not posted on weakpass.com can be found at https://github.com/FullTang/AndroidPWList. This list combines several of the smaller wordlists from Weakpass and uses a few other techniques for an uncompressed size that is just under 1GB in size. If you plan on installing and using Hashcat, I would strongly recommend downloading at least one list of your choice.

Installing Hashcat

Now that we know some of the more common methods used to create passwords, and we have access to a good list of millions of potential passwords, we can attempt to crack the example hash using Hashcat. The most recent version of Hashcat can be securely downloaded at https://hashcat.net/hashcat/ (Hashcat - Advanced Password Recovery, n.d.). Considering the type of calculations performed, it is much more efficient to use the video card of a computer to perform these calculations rather than use the CPU. This may cause some compatibility issues, and fixing those issues is beyond the scope of this writing. I would encourage anyone who has not used Hashcat or even if they have not used a command-line tool to follow along at this point on their own Windows machine even if you have not extracted any hashes up to

this point. We will crack the previously mentioned example hash (b4b9b02e6f09a9bd760f388b67351e2b) from Hashcat's website here shortly!

Once Hashcat is installed, it needs to be launched from the command line, or command prompt, assuming the user is using a Windows system. The simplest method to launch a command prompt window in the correct location is to navigate to the where Hashcat is installed (C:WindowsPrograms\hashcat-6.2.5 or similar) using File Explorer, click the white area next to the path so that the path turns blue, type "cmd" without the quotes, and press enter. A black window with white text should appear. If you have never used the command line before, congratulations on opening your first terminal window!

The next step is to launch Hashcat in help mode. This will also see if the correct drivers are installed to allow for Hashcat to run. Simply type "hashcat.exe -h" in the command line without the quotes. It is possible that an error occurred stating an OpenCL, HIP, or CUDA installation was not found. If this is the case, I would recommend typing "device manager" in the search bar next to the Windows Start menu and then selecting "Display adapters" to determine the type of video card installed on the computer. Beyond this, it will require downloading the required drivers from a trusted source to continue using Hashcat. Additional help on how to install Hashcat can be found on the Hashcat Discord server at the following link: https://discord.gg/vxvGEMuemw.
If the "hashcat.exe -h" is successful, then there should be a large amount of output on the screen showing options, hash modes, and examples, and should end with some links to the Hashcat website. I find it helpful to save this help information to a simple text file for easy reference. That can be done by pressing the up arrow on the keyboard to display "hashcat.exe -h" again, but before pressing enter add "> Help.txt" to the end of the command for the total command of "hashcat.exe -h > Help.txt". This will create a text file in the same folder with the output from the help command

which can be opened in Notepad or similar for quick reference while keeping the command prompt window free to run Hashcat.

Open the Help.txt that was just created in the hashcat-6.2.5 folder. Under - [Hash Modes] - it shows the numerous types of hashes that can be attacked (and possibly cracked) assuming the hash is properly extracted. Scrolling to the bottom shows some example commands to run Hashcat under - [Basic Examples] -. Note that the first Attack-Mode is a Wordlist, but there is also a Brute-Force option. This is not a true brute force method as was discussed earlier as it does not use all the possible symbols on the keyboard nor does it use uppercase letters except for the first character. One advantage is that it does not require a dictionary or wordlist to crack a password, so it has its uses. Let's break down this command.

Under example command, the first word is "hashcat". It can also be "hashcat.exe". This is simple, we are just calling the executable file, but we need to give some input or arguments to the program. The next thing we see is "-a" and then a number followed by "-m" followed by another number. At the top of the help file, we see under - [Options] - it explains "-a" as the attack-mode and "-m" as the hash-type. Both of these are required, but the order is not an issue as they be in either order, but we will follow the order shown in the example. Scrolling back down towards the bottom we find - [Attack Modes] - where it shows the types of attacks. Brute-Force is 3 while Straight is 0. Brute-Force is Hashcat's version of brute-force that was just discussed, while Straight is a dictionary attack using a wordlist. Now for the other required argument, the "-m". This stands for hash-type, so we scroll up to the bulk of the help file under - [Hash Modes] - and see all the different types. We know this is an NTLM hash, so we need to find the hash-type for NTLM in all of that noise. Rather than manually searching, press CTRL + F to open the find menu and type "NTLM" (without the quotes). You may get some results like "NetNTLMv1", "NetNTLMv1+ESS", or "NetNTLMv2" and you may have to change your direction of searching to find matches, but you should be able to find just NTLM

all on one line with a mode of 1000. Now that we know the required parameters for our two required arguments, onto how to input the hash itself into Hashcat.

When it comes to the hash itself, Hashcat will accept the hash in one of two ways. It can either be pasted directly into the command line, or it can be put into a simple text file (.txt) with one hash (and only one hash) per line. If a text file is used, it needs to be all hashes of the same type, like multiple NTLM hashes or multiple SHA-256 hashes, with each hash on its own line. If attacking multiple hashes, the file method will be faster than trying to crack them one at a time but it will be slower than a single hash. Pasting directly into the command line can be faster if the hash is already extracted, but a few seconds taken to format the hash in a text file right after extraction may be better in some situations.

The example hash shows some arguments like ?a?a?a?a?a? after the "example0.hash", but those are not required. Other arguments can be seen towards the top of the help file, but those are optional. We now know everything required to crack this example NTLM hash! (b4b9b02e6f09a9bd760f388b67351e2b).

"Brute-Forcing" with Hashcat

Go to the command line where we typed in hashcat.exe -h and type hashcat.exe -a 3 -m 1000 b4b9b02e6f09a9bd760f388b67351e2b and hit enter. There should be a wall of white text and then it will stop and it should show "Cracked" partway up on the screen! Above the Cracked notification, there will be the hash and at the end, it will show ":hashcat" or "b4b9b02e6f09a9bd760f388b67351e2b:hashcat" without the quotes. This means the password was hashcat, as can be seen at the top of the Hashcat Wiki webpage. If this is your first time cracking a password then congratulations! You just cracked your first password hash! Now let's examine what Hashcat did during that wall of white text.

Scrolling up we can see the first block of text similar to the block of text at the end, but instead of saying "Cracked" it says "Exhausted". Looking at the "Guess.Mask" row in the first column we see a "?1 [1]", and on the next row we see a "Guess.Charset". On the "Guess.Charset" row there it shows the -1 and it is followed by a ?l?u?d. To know what those mean, we need to go back to our help file. Under - [Built-in Charsets] - close to the bottom we see the "l" showing all lowercase characters, the "u" showing all uppercase characters, and the "d" is all digits from 0 to 9. Putting it all together this means Hashcat tried all lowercase, uppercase, and digits for a password length of 1 before exhausting and moving on. Notice how at the top it showed "Approaching final keyspace - workload adjusted." and that means that Hashcat realizes it is about to come to the end of its current process and it is thinking about what it needs to do next.

The second block shows a Guess.Mask of ?1?2 [2]. Therefore, there was a total of two characters, but this time it is a little different. The ?2 is only the ?l and ?d meaning for the second character it only tried lowercase and digits, but for the first character it was still a ?1 so it tried lower, upper, and digits like in the first block. The third block is a Guess.Mask of ?1?2?2 [3], so three characters total but only trying uppercase, lowercase, and digits for the first and trying lowercase and digits for the other two. The fourth, fifth, and sixth blocks all show uppercase, lowercase, and digits for the first character with lowercase and digits for the rest. The seventh block is where it was cracked, using the same Guess.Mask format of ?1?2?2?2?2?2?2. The password was not long enough to see for this example, but if we didn't crack it on seven characters it would keep getting longer, and eventually the ?3 would be used which would be added to the end which would also try five symbols, not the parentheses, of (*!$@_) in addition to lowercase and digits for the last character.

Hashcat's Potfile

This worked for this password, but for more complicated passwords we can see where it has its limitations. That is why we need a robust wordlist. So let's try and crack this password again using a wordlist, and in doing so we will discover a useful function of Hashcat. First, find the wordlist that you previously downloaded in file explorer and unzip it. It may not have a file extension, but Hashcat doesn't care nor would it be likely that you could open the file in normal Notepad anyway as it is probably going to be too big for the standard version of Notepad. If you want to see the contents, you should be able to use another text editor like Notepad++ for smaller wordlists, but it is by no means required. Let's go back to the command line where we just cracked the hash and type out a new command. Type "hashcat.exe -a 0 -m 1000 b4b9b02e6f09a9bd760f388b67351e2b " not forgetting to put a single space after the hash but don't hit enter just yet. Hashcat needs the path for the wordlist, note how we are using -a 0 instead of -a 3. If you are savvy with the command line, you could enter the path of the file (not forgetting quotes if there are any spaces), or you could copy the path from the file explorer window (where we typed "cmd" earlier to open our command prompt window) and then add the file name, but there is an easier way that some may consider cheating. If you are not cheating you are not trying, right? The easiest way is to just drag and drop the uncompressed wordlist into the black area of the command prompt window and it should populate the whole path to the file in the command line. The whole command should look something like this, "hashcat.exe -a 0 -m 1000 b4b9b02e6f09a9bd760f388b67351e2b "D:My FolderMy Downloaded Wordlist"". There may or may not be quotes around the path depending on if there are spaces in the folder and subfolders or the file name. Hit enter and see what happens.

It should have finished very quickly and displayed a notification of "INFO: All hashes found in potfile! Use –show to display them.".

Well, that is interesting, what is a potfile? Simply put, the potfile is where Hashcat automatically stores hashes it cracks with the corresponding password in plain text. This is very useful to make sure that time is not wasted trying to crack passwords that have already been cracked and to make sure a cracked password is saved in case of power failure. It would be most unfortunate if a password was cracked before the examiner could see it and the power went out to the machine that was not hooked up to a Universal Power Supply due to budgetary concerns. Anyway, go to the hashcat-6.2.5 folder where hashcat.exe is located, find the file named "hashcat.potfile" and open using Notepad or the text editor of your choice. Assuming this is your first time using a freshly downloaded Hashcat, there will only be one entry, "b4b9b02e6f09a9bd760f388b67351e2b:hashcat. This is nice to prevent us from wasting time trying to crack it again, but we want to see how to try and crack it using other methods. Either delete the single entry from the potfile, save, and close, or just delete the whole potfile as Hashcat will automatically generate a new one upon cracking another password.

Dictionary (Wordlist) Attack with Hashcat

Go back to the command prompt and press the up arrow on the keyboard. Your previously typed command of "hashcat.exe -a 0 -m 1000 b4b9b02e6f09a9bd760f388b67351e2b "D:My FolderMy Downloaded Wordlist"" or similar should appear. Press Enter to run the command again. Now it should start processing, but it will stop after a moment and display something like "Watchdog: Temperature abort trigger set to 90c". As a side note, this is nice to know that Hashcat has built-in safety procedures to help prevent the overheating of video cards and will slow down its processing speed if the GPU (aka video card) gets too hot. Anyway, after a few

seconds, it should display something like "Dictionary cache building "D:My FolderMy Downloaded Wordlist": 1711225339 bytes (10.61%)" with the percentage increasing every few seconds. This is normal and depending on the size of the wordlist it might take a minute or two. This is required after the first time starting a new wordlist, but as long as the location of the wordlist does not change it will not need to build the dictionary each time. Once the dictionary is built, it will display the following line: "[s]tatus [p]ause [b]ypass [c]heckpoint [f]inish [q]uit ⇒" This shows what commands we can enter while it is processing. It would be nice to know what is going on, so press the 's' key.

The first thing I look at is the Time.Estimated row and it will show an estimated end date and time and estimated duration. This is where times can vary greatly based on the type of GPU and length of the wordlist. Even if a longer wordlist was chosen, it should not take long to crack the password This is assuming that the word "hashcat" is in the dictionary, but hopefully it is there. This method will likely take a bit longer than the brute-force method, but it is much more robust and is one of the best methods for cracking passwords. We are going to try one more method for now, so go back to the potfile and delete the most recent entry from the potfile or just delete the whole potfile.

Dictionary + Rules with Hashcat

The obvious weakness of the dictionary attack is the password has to be in a precompiled dictionary, but what if it is a complicated password not in a wordlist? What if the user made a password that used unusual symbols or used numbers at the beginning, used numbers instead of letters, or added an unusual number of symbols to the end? This can be cracked by Hashcat by using a combined dictionary and rule attack. Hashcat comes preloaded with rules, and additional rules can be downloaded just like wordlists can

be downloaded. At this time, I have not found any rules that are noticeably superior to the rules that come standard with Hashcat but it is left up to the examiner to decide what they want to use.

After deleting the most recent entry in the potfile, check the "hashcat-6.2.5" folder and there should be a folder named "rules". Inside the rules folder, there are plenty of prebuilt rules. My personal favorite is the "onerulestorulethemall" rule as the name rings home to me because of my short stature. It is also a good rule in general, but again there is mostly personal preference. It is worth mentioning that while these rules are only a few kilobytes in size, they can add a substantial amount of time to how long it takes to process a hash as all commands in each rule will be applied to each potential password in a wordlist. Just like with dictionary attacks, a bigger rule will take longer and yield more potential passwords but a smaller rule will be faster but with fewer generated passwords.

Go back to the command prompt and press the up arrow. Adding a rule to a dictionary attack is quite easy, we just need to add a "-r" followed by the path to the rule file after the dictionary at the end of the command. Just add -r to the end of the command, put a space, then drag and drop the rule of your choice into the command prompt window. The command should look something like "hashcat.exe -a 0 -m 1000 b4b9b02e6f09a9bd760f388b67351e2b "D:My FolderMy Downloaded Wordlist" -r D:\hashcat-6.2.5\rules\onerulestorulethemall.rule"". Once syntax looks good, press enter. This time the dictionary should not have to compile, as it will display "Dictionary cache hit:" and then information on the location of the dictionary. Press the "s" key on the keyboard to see the status, and note how the "Time.Estimated" row has increased, possibly to a day or more. Hopefully, it will not take longer than a few minutes to crack our example hash again. This method does take longer, but again we are attacking the hash in a way that will crack more complicated passwords than the other methods.

Robust Encryption Methods

Up to now we have only cracked an NTLM hash, but what about more robust encryption methods? Go to the Hashcat Example Hashes webpage (https://hashcat.net/wiki/doku.php?id=example_-hashes) and search for "Bitlocker" that should be mode 22100. The resulting hash should be: "$bit-locker$1$16$6f972989ddc209f1eccf07313a7266a2$1048576$12$3a33a
8eaff5e6f81d907b591$60$316b0f6d4cb445fb056f0e3e0633c413526ff4481bb
f588917b70a4e8f8075f5ceb45958a800b42cb7ff9b7f5e17c6145bf8561ea86f52d3592059fb
This is massive compared to the NTLM hash! Try it in Hashcat using the command "hashcat.exe -a 3 -m 22100 $bit-locker$1$16$6f972989ddc209f1eccf07313a7266a2$1048576$12$3a33a
8eaff5e6f81d907b591$60$316b0f6d4cb445fb056f0e3e0633c413526ff4481bb
f588917b70a4e8f8075f5ceb45958a800b42cb7ff9b7f5e17c6145bf8561ea86f52d3592059fb
without the quotes. The brute-force starts at four characters, because BitLocker originally required a minimum password length of four so Hashcat is smart enough to not waste time with trying less than four characters when attacking a BitLocker password. For my computer it shows an estimated time of 1 hours 19 minutes for just 4 characters! If I let it run and go to 5 characters, it shows it will take 2 days to just try 5 characters! Your computer may have different estimated times, but unless you have a really good gaming computer or are running Hashcat on a computer designed for mining cryptocurrency you are probably seeing similar numbers. Trying the same Bitlocker hash but just using a dictionary attack with no rules against the cyclone.hashesorg.hashkiller.combined dictionary shows an estimated time of 28 days!

Knowing this means that if an NTLM hash was cracked using the cyclone.hashesorg.hashkiller.combined dictionary, it will take about a month at the most for the same Bitlocker password to be cracked This time can be significantly reduced by using a computer with multiple GPUs like computers used for mining cryptocurrency. This is a really good reason to not have a password that comes

standard in most dictionary attacks and shows why strong and complicated passwords are important.

This is just examining Bitlocker, but VeraCrypt and DiskCryptor example hashes require the download of a file as it is too large to display on Hashcat's website. This shows a substantial difference between password encryption used by Windows and robust encryption software, but it also shows why it is very important to not reuse passwords. If an attacker can compromise the weak Windows password and it is the same password that is used for robust encryption software then it defeats the strong encryption method. It also shows how a robust encryption method can be defeated by a good wordlist and shows why strong passwords are the first line of defense no matter what encryption method is used.

Complex Password Testing with Hashcat

Maybe you have gotten the bug by now and our simple hash that is just "hashcat" is not good enough and you want to try even harder potential passwords. The easiest way to attempt to crack more difficult passwords is to use an NTLM hash generator. Online NTLM hash generators hosted on a website may be the easiest route, but there is a major security concern if the user wants to test their own passwords and converts them using an online tool. By using the online tool the user has likely given up their password to a third party if that online tool is logging input to their website. I would only recommend using an online tool for testing passwords that the user is not using, and I would not even use similar passwords to ones that are currently in use in an online tool.

The next best method would likely be PowerShell functions or Python scripts that can generate NTLM hashes. Searching Google

can find these functions and scripts, but the methods for using them will not be discussed. This is much more secure as the processing to convert the password to an NTLM hash is done on the user's computer. Just note that if the password is cracked, it will be saved in the potfile so it would be wise to either delete the entry from the potfile or delete the potfile altogether once the testing session is complete.

Searching a Dictionary for a Password

Since we have already mentioned that the main weakness of a password is it's existance in a wordlist, it might be nice to see if our current password or other potential password shows up in a dictionary. Since these wordlists are very large, it is difficult to find a program that will open them up to do a simple Ctrl + F to search the document to find the password. Fortunatally, the command line offers and easier way to search the contents of a file without opening the file. Using file explorer, navigate to the folder where you have downloaded and uncompressed a wordlist. Openup a command line window just like we did for running Hashcat by clicking the white area next to the path so that the path turns blue, type "cmd" without the quotes, and press enter. A black window with white text should appear. We are going to use the "findstr" command to search the contents of a dictionary. In the command line, type "findstr password " and then press [TAB] until the dictionary you want to search appears. The completed command should look somthing like "findstr password MyDiscitonary" without the quotes. Press enter. If you chose a common password it should output a wall of white text showing all passwords with that contain that password. If it just shows a blinking cursor, then it is searching trying to find a match. When you are able to type again, it has finished searching.

This is a good way to check if a password exists in a dictionary or wordlist, but if the password does not show up that does not mean it can't cracked with that dictionary, but an approprate rule would have to be added in order to mangle the wordlist in a way that would cause the password to be guessed by Hashcat. Still, since dictionary attacks are the most common and the fastest method of cracking a password, it is a good yet simple test to see if the password is a strong password or not.

Generating Custom Wordlists

Paring Down Custom Wordlists

Mangling Custom Wordlists

Conclusion

This has just been a brief dive into showing how easy it is to crack simple passwords and hopefully will show why strong passwords are so important. The Windows operating system uses a weak form of encryption for its passwords, and this is a place to start when trying to crack passwords for fun or security testing purposes. Even with strong encryption methods, a weak password will not be sufficient to safeguard the data. Knowing these methods are out there to defeat user passwords should show the user why it is so important to use strong passwords and why it is a bad idea to reuse passwords between accounts. A better understanding of the attack methods against passwords should encourage everyone to use better security practices to safeguard their data.

References

Cubrilovic, N. C. (2009, December 14). TechCrunch is part of the Yahoo family of brands. TechCrunch. Retrieved May 12, 2022, from https://techcrunch.com/2009/12/14/rockyou-hack-security-myspace-facebook- passwords/

hashcat - advanced password recovery. (n.d.). Hashcat. Retrieved May 12, 2022, from https://hashcat.net/hashcat/

Patton, B. (2022, March 25). NTLM authentication: What it is and why you should avoid using it. The Quest Blog. Retrieved May 12, 2022, from https://blog.quest.com/ntlm-authentication-what-it-is-and-why-you-should-avoid-using-it/

Picolet, J. (2019). Hash Crack: Password Cracking Manual (v3). Independently published.

W34kp455. (2014). Weakpass. Weakpass. Retrieved May 12, 2022, from https://weakpass.com/

Section 2 - Mobile Forensics

Chapter 5 - Chapter Title Goes Here

Subheading goes here

Content goes here. Remember, only what is put here is parsed by Leanpub to generate the PDF/MOBI/EPUB files.

Chapter 6 - Chapter Title Goes Here

Subheading goes here

Content goes here. Remember, only what is put here is parsed by Leanpub to generate the PDF/MOBI/EPUB files.

Chapter 7 - Chapter Title Goes Here

Subheading goes here

Content goes here. Remember, only what is put here is parsed by Leanpub to generate the PDF/MOBI/EPUB files.

Chapter 8 - Chapter Title Goes Here

Subheading goes here

Content goes here. Remember, only what is put here is parsed by Leanpub to generate the PDF/MOBI/EPUB files.

Chapter 9 - Gamification of DFIR: Playing CTFs

By Kevin Pagano[17] | Stark 4N6[18]

What is a CTF?

The origins of CTF or "Capture The Flag" were found on the playground. It was (still is?) an outdoor game where teams had to run into the other teams zones and physically capture a flag (typically a handkerchief) and return it back to their own base without getting tagged by the opposing team. In the information security realm it has come to mean a slightly different competition.

[17] https://twitter.com/KevinPagano3
[18] https://startme.stark4n6.com

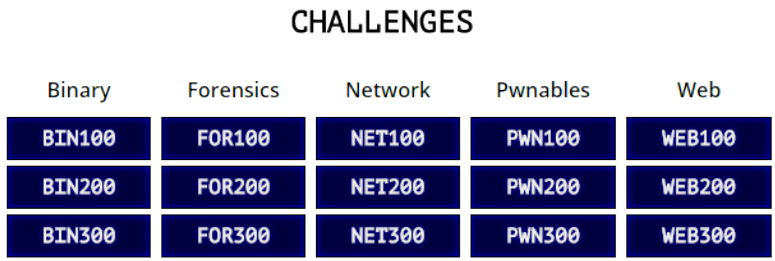
Why am I qualified to talk about CTFs?

Humble brag time. I've played in dozens of CTF competitions and have done pretty well for myself. I am the proud recipient of 3 DFIR Lethal Forensicator coins from SANS, one Tournament of Champions coin (and trophy!), a 3-time winner of Magnet Forensics CTF competitions, a 4-time winner of the BloomCON CTF competition, and a few others. I've also assisted in the creation of questions for some CTF competitions as well as creating thorough analysis write-ups of events I've competed in on my personal blog[19].

Types of CTFs

Two of the most common information security types of CTF competitions are "**Jeopardy**" style and "**Attack and Defense**" style.

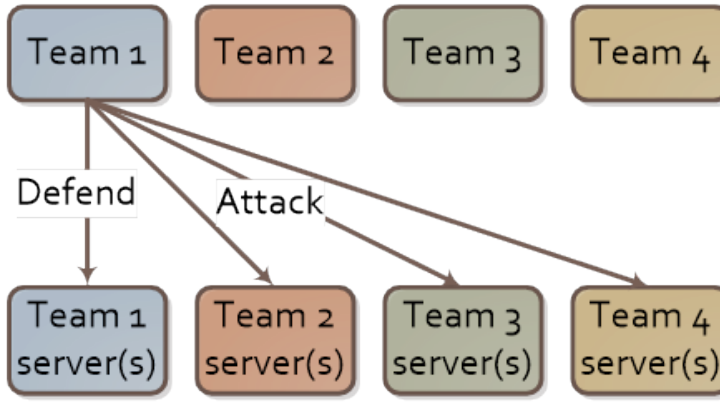
"**Jeopardy**" style typically is a list of questions with varying difficulty and set defined answers. The player or team is given some sort of file or evidence to analyze and then has to find the flag to the question and input it in the proper format to get points.



9.1 - Jeopardy style CTF

[19] https://ctf.stark4n6.com

"**Attack and Defense**" is more common in Red and Blue team environments where the Red team has to hack or attack a Blue team server. The Blue team subsequently has to try and protect themselves from the attack. Points can be given for time held or for acquiring specific files from their adversary.



9.2 - Attack and Defense CTF

Depending on the CTF, you may see a combination of types with it being Jeopardy style and linear (story based) with some questions hidden or locked until a certain question is answered.

For this chapter, I will go more in-depth regarding the "Jeopardy" style competitions, more specifically, forensics geared CTF competitions.

Evidence Aplenty

With forensics CTF's, just like in real life, any type of device is game for being analyzed. In the ever growing landscape of data locations, it just provides us more places to look for clues to solve the problems. One of the more well known forensic CTF's is the

SANS NetWars[20] tournaments. These are devised with 5 levels with each level being progressively harder than the last. In this competition you will have a chance to analysis evidence from:

- Windows computer
- macOS computer
- Memory/RAM dump
- iOS dump
- Android dump
- Network (PCAP/Netflow/Snort logs)
- Malware samples

You can see from the above list that you get a well rounded variety of types of evidence that you most likely will see in the field on the job. In other competitions I've played you could also come across Chromebooks or even Google Takeout and other cloud resources as they become more common. I have also seen some where they are more crypto based so working with different ciphers and hashes to determine the answers.

Who's Hosting?

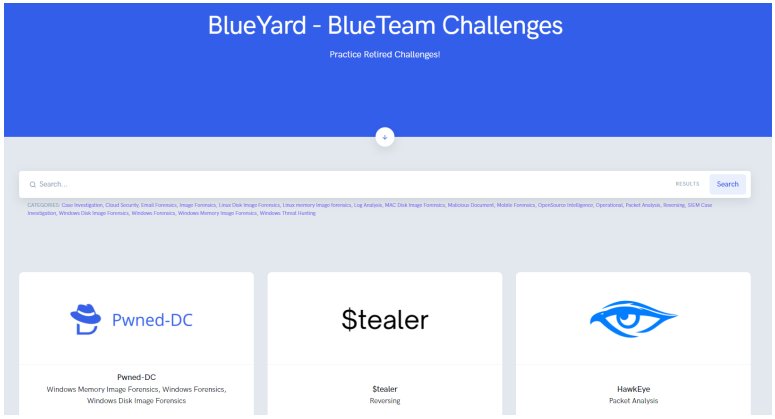
As previously mentioned, **SANS** is probably the most well known provider of a forensics CTF through their NetWars program. It isn't cheap as a standalone, but is sometimes bundled with one of their training courses. You can sometimes see them hosted for free with other events such as OpenText's enFuse conference.

As for others, **Magnet Forensics** has been hosting a CTF for the past 5 years in tandem with their user summit. This has been created by Jessica Hyde in collaboration with some students from Champlain College's Digital Forensics Association. Some previous Magnet CTF's were also created by Dave Cowen and Matthew Seyer for context.

[20] https://www.sans.org/cyber-ranges/

Other software vendors have started to create their own as well to engage with the community. **Cellebrite** in the past 2 years has hosted virtual CTF competitions and **Belkasoft** has created and put out multiple CTFs the last 2 years. **DFRWS**[21] hosts a yearly forensic challenge with past events covering evidence types such as Playstation 3 dumps, IoT (Internet of Things) acquisitions, mobile malware, and many others.

Another fantastic resource for finding other challenges is **CyberDefenders**[22]. They host hundreds of various different CTF challenges, from past events and other ones that people have uploaded. You can even contribute your own if you'd like as well as allow them to host your next live event.



9.3 - CyberDefenders website

Another fairly exhaustive list of other past challenges and evidence can be found hosted on **AboutDFIR**[23].

[21] https://dfrws.org/forensic-challenges/
[22] https://cyberdefenders.org/blueteam-ctf-challenges/
[23] https://aboutdfir.com/education/challenges-ctfs/

Why Play a CTF?

So at the end of the day, why should YOU (yes, YOU, the reader) play a CTF? Well, it depends on what you want to get out of it.

For Sport

Growing up I've always been a competitive person, especially playing sports like baseball and basketball, CTF's are no different. There is a rush of excitement (at least for me) competing against other like-minded practitioners or analysts to see how you stack up. You can even be anonymous while playing. Part of the fun is coming up with a creative handle / username to compete with. It also keeps the commentary and your competitors on their toes.

I personally like to problem solve and to be challenged which is part of the reason why I enjoy playing.

For Profit

I put profit in quotations because many may construe that as a compensation type objective. While many CTF challenges do have prizes such as challenge coins or swag (awesome branded clothing anyone?!) that's not completely the profit I'm talking about here. The profit is the knowledge you gain from playing. I've done competitions where I never knew how to analyze memory dumps at all and I learned at least the basics of where to look for evidence and new techniques to try later on in real world scenarios.

> "Commit yourself to lifelong learning. The most valuable asset you'll
> ever have is your mind and what you put into it." - Albert Einstein

The knowledge you gain from the "practice" will inevitably help you in the future, it's just a matter of time. Seriously, you don't know what you don't know. Remember when I said you can be anonymous? It doesn't matter if you get 10 points or 1000 points, as long as you learn something new and have fun while doing so, that's all that matters.

Toss a Coin in the Tip Jar

I get asked all the time, "what are your keys to success playing CTFs?". That's probably a loaded question because there are many factors that can lead to good results. Here I will break down into sections that I feel can at least get you started on a path forward to winning your first CTF.

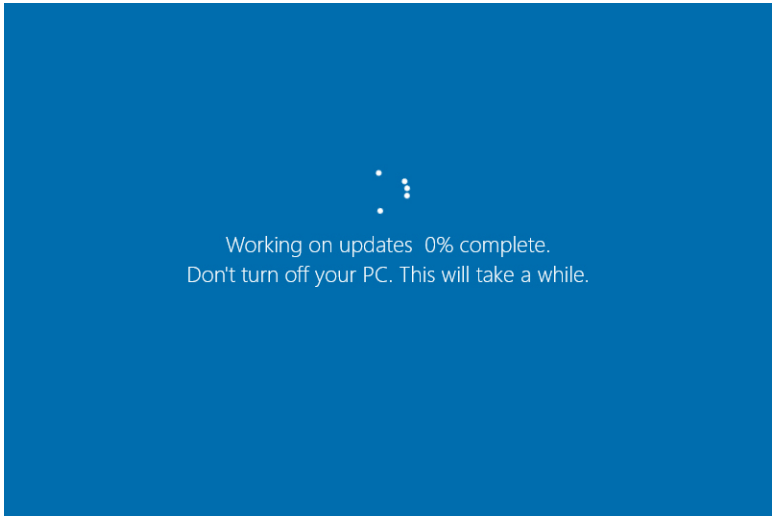
Tips for Playing - Prior

First and foremost is the preparation phase. Like any task in life, it always helps to be prepared for the battle ahead. Having a sense of what is to come will help with your plan of attack. Do your research! If you know that a specific person created the CTF then take a look at their social media profiles. Often times they will release hints in some form or fashion, whether it is webinars they have shared or research papers or blog posts they have recently published. Don't over do it though, there could be red herrings amuck. You can also look at past CTF's they have created, seeing how questions were formulated before and what sort of locations they tend to lean on for flags. This is part of the reason why I personally do write-ups of past CTF's for future reference.

Each CTF rules may be different but if teams are allowed reach out to colleagues or others to form a squad. Knowledge from multiple people well-versed in different topics can help in spreading out the

workload especially if there are multiple forms of evidence to be analyzed. I would be remiss if I didn't say that some of my winning efforts were with team members who helped pick up sections where I wasn't as strong. Your mileage may vary though, make sure to coordinate your efforts if you do join a team as to not waste time all working on the same questions.

If evidence is provided ahead of the competition make sure to spend some time getting familiar with it. Process the evidence ahead of time so you aren't wasting time during the live competition waiting on machine time. Some of these events only last 2-3 hours so time is of the essence. This segues right into building out your analysis machine and your toolkit. Make sure that all your system updates are completed prior. The last thing you need is an errant Windows update to take down your system while you watch the spinning.



9.4 - "This will take a while"

You may also consider making sure you have local admin access or at least the ability to turn of antivirus (if you are analyzing malware) to your computer. Always do so in a controlled environment if possible but you knew this already (I hope). If you are provided a

toolkit or a trial of a commercial license, use it to your advantage, even if it's a secondary set of tools. There are times some vendors will make sure that the answer is formulated in a way that their tool will spit out from their own software. Also, commercial tools can potentially speed up your analysis compared to a bunch of free tools but that is personal preference.

The Toolkit

I'm a Windows user through and through so I cannot offer much advise from a Mac or Linux perspective. With that said, I do have some tools that I use from a forensic perspective to analyze those types of evidence. Here are my favorite (free) tools that I use during CTF's:

General Analysis

- Autopsy[24]
- Bulk Extractor[25]
- DB Browser for SQLite[26]
- FTK Imager[27]
- Hindsight[28]

Chromebook

- cLEAPP[29]

Ciphers

- CyberChef[30]
- dcode.fr[31]

[24] https://www.autopsy.com/
[25] https://github.com/simsong/bulk_extractor
[26] https://sqlitebrowser.org/dl/
[27] https://www.exterro.com/ftk-imager
[28] https://dfir.blog/hindsight/
[29] https://github.com/markmckinnon/cLeapp
[30] https://gchq.github.io/CyberChef/
[31] https://www.dcode.fr/en

Google Takeout / Returns
- RLEAPP[32]

Mac
- mac_apt[33]
- plist Editor - iCopyBot[34]

Malware/PE
- PEStudio[35]
- PPEE (puppy)[36]

Memory/RAM
- MemProcFS[37]
- Volatility[38]

Mobile Devices
- ALEAPP[39]
- Andriller[40]
- APOLLO[41]
- ArtEx[42]
- iBackupBot[43]
- iLEAPP[44]

Network
- NetworkMiner[45]
- Wireshark[46]

[32] https://github.com/abrignoni/RLEAPP
[33] https://github.com/ydkhatri/mac_apt
[34] http://www.icopybot.com/plist-editor.htm
[35] https://www.winitor.com/
[36] https://www.mzrst.com/
[37] https://github.com/ufrisk/MemProcFS
[38] https://www.volatilityfoundation.org/releases
[39] https://github.com/abrignoni/ALEAPP
[40] https://github.com/den4uk/andriller
[41] https://github.com/mac4n6/APOLLO
[42] https://www.doubleblak.com/software.php?id=8
[43] http://www.icopybot.com/itunes-backup-manager.htm
[44] https://github.com/abrignoni/iLEAPP
[45] https://www.netresec.com/?page=NetworkMiner
[46] https://www.wireshark.org/

Windows Analysis
- Eric Zimmerman tools / KAPE[47]
- USB Detective[48]

This whole list could be expanded way further but this is the majority of the go-to's in my toolkit.

Tips for Playing - During

We've all been there, you get to a point in the middle of a CTF and you start to struggle. Here are some things to key in on while actually playing.

Read the titles of the questions carefully, often times they are riddled with hints about where to look. "*Fetch"* the run time of XXX application, maybe you should analyze those Prefetch files over there. Questions will often also tell you what format the answer should be in when submitting. This may tell you that that timestamp you're hunting could be incorrect, those pesky timezone offsets!

Did you find a flag that appears to be a password? It's almost guaranteed that that evidence was placed in such a way that it will be reused. Emails and notes can be a treasure trove for passwords to encrypted containers or files.

One thing that may seem silly but can help is to just ask questions. If you're stumped on a question, talk to the organizer if you can, they may lead you in a direction that you didn't think of when you set off on a path of destruction.

Some CTF competitions have a built in hint system. If they don't count against your overall store, take them! The chances of a tie breaker coming down to who used less hints is extremely small. If the hint system costs points you will need to weigh the pros and

[47] https://ericzimmerman.github.io/#!index.md
[48] https://usbdetective.com/

cons of not completing a certain high point question as opposed to loosing 5 points for buying that hint.

The last tip while playing is to write down your submissions, both the correct and incorrect ones. I can't tell you the amount of times I've entered the same answer wrongly into a question to eventually get points docked off my total. This will not only help you during the live CTF but afterwards if you are writing a blog on your walkthroughs.

Strategies

Takeaways

Section 3 - Computer Forensics

Chapter 10 - Getting into Digital Forensics

By Joshua I. James[49] | DFIR Science[50]

v220405

[49] https://www.youtube.com/c/DFIRScience
[50] https://dfir.science

So you want to be a digital forensic investigator?

Technical Skills

How to build a home lab?

Non-Technical Skills

WRITING

PRESENTING

Common Career Paths

Programming

Languages

Tools

Cooperation and Collaboration

Why Collaborate?

How to Collaborate?

Research

Conferences

Online Resources

International Cooperation

Informal

Formal

Chapter 11 - Chapter Title Goes Here

Setting Up a Law Enforcement Digital Forensics Laboratory

By Jason Wilkins MCFE, 3CE[51]

[51] https://twitter.com/TheJasonWilkins

Executive Cooperation

The necessity of executive cooperation

Making your case to executive leadership

Open communication and trust

Physical Requirements

Physical security and accessibility

Floor plans

Selecting Tools

Network Requirements

Selecting forensic workstations

Selecting forensic software

Selecting peripheral equipment

Planning for Disaster

Certification and Training

Why should you get certified?

Where to find training

Creating a training plan for your lab

Accreditation, Policy, and Procedure

Accreditation requirements

Chapter 12 - Chapter Title Goes Here

Subheading goes here

Content goes here. Remember, only what is put here is parsed by Leanpub to generate the PDF/MOBI/EPUB files.

Section 4 - Title Goes Here

Chapter 13 - Chapter Title Goes Here

Subheading goes here

Content goes here. Remember, only what is put here is parsed by Leanpub to generate the PDF/MOBI/EPUB files.

Chapter 14 - Chapter Title Goes Here

Subheading goes here

Content goes here. Remember, only what is put here is parsed by Leanpub to generate the PDF/MOBI/EPUB files.

Chapter 15 - Chapter Title Goes Here

Subheading goes here

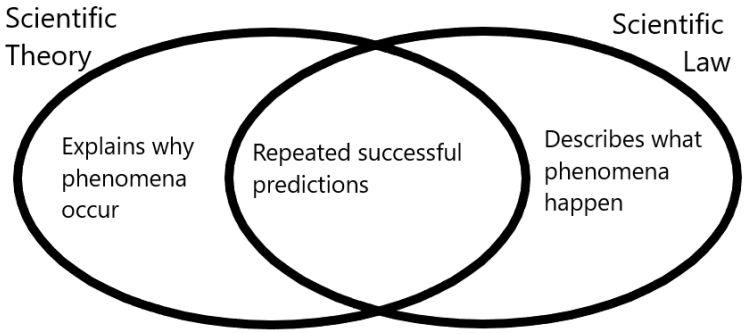
Content goes here. Remember, only what is put here is parsed by Leanpub to generate the PDF/MOBI/EPUB files.

Chapter 16 - Artifacts as Evidence

Forensic Science

Before learning about artifacts as digital evidence, I'll preface this chapter with the most fundamental definition of basic science. So what is Science? It is a field that follows a scientific process in any of it's domain. That process is cyclical and goes something like this:

- We make observations in nature.
- We form an initial hypothesis about something.
- We look for things that confirm or deny the formed hypothesis.
- If we find something that denies it, we form a new hypothesis and go back to making observations.
- If we find something that confirms it, we continue making new observations to extend our dataset and verify the hypothesis until the dataset is substantial in confirming it precisely and accurately. If we further find something that denies the original hypothesis, we form a new one repeating the process.



16.1 - Attr: GliderMaven, CC BY-SA 4.0, via Wikimedia Commons

Ultimately, the goal of any science is not to state things in absolutes, but in observations, experiment, procedure and conclusions. Even the fundamental laws of science begin with a basic foundation laid of assumptions.

Much like any scientific field, forensics or criminalistics is a branch of science that deals with identifying, collecting and preserving evidence of a crime. It is not just identifying, collecting and preserving but doing so in a forensically sound manner. What that means is that evidence should not be changed or stray away from it's true form.

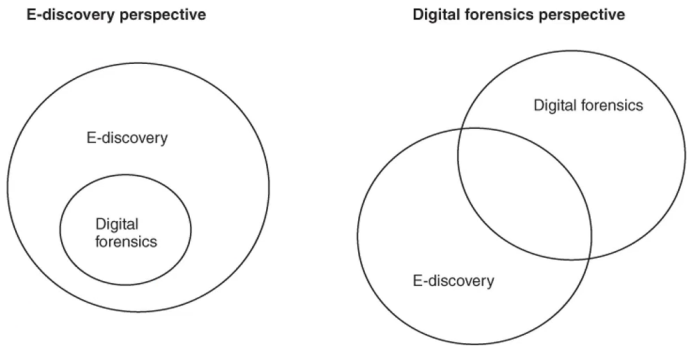
Digital Forensics is a six phase process including Preparation, Identification, Collection, Preservation, Analysis and Reporting.

Types of Artifacts

Analysis is a major phase where in forensicators discover different types of artifacts ranging from plain metadata to complex evidence of execution and residual traces. The vast gap between the difficulty to retrieve or reconstruct evidence determines the fine line between E-discovery and Digital Forensics.

User data such as internet history, images, videos, emails, messages etc fall under E-discovery. It is relatively easy to reconstruct even from the unallocated space.

However, System Data like artifacts that help support some view of truth, or determine how closely a transpired event is to the evidence, are not that simple to manually parse with forensic soundess, which is why oftentimes forensicators rely on well-known parsing tools either commercial or opensource.



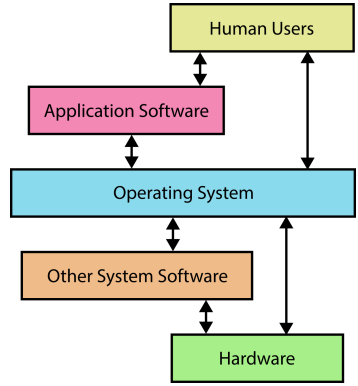
16.2 - Attr: Slide 6 from E-Discovery: An Introduction to Digital Evidence by Amelia Phillips, Ronald Godfrey, Christopher Steuart & Christine Brown

And that is the main difference between E-Discovery & Digital Forensics depending on the categorization of data alone. Both follow different procedures and have different scope of execution. Generally, E-Discovery can be contained to only the logical partitions and the unallocated region whereas Digital Forensics operates in a much wider scope solely due to the necessity of dealing with complex data structures.

What is Parsing?

Which brings us to parsing. We often go around throwing the term while working with a variety of artifacts; “Parse this, parse that”, but what does it mean in the real sense? To understand the parsing methodology, tools & techniques, we must be familiar with the origin of the handling of the data being parsed. What I mean by that is how was the data originally meant to be handled. What was it’s structure by design. How can it be replicated.

Generally, it is some feature or underlying mechanism of the main operating system installed on the device. Parsing tools are written to accurately mimic those functions of the operating system which make the raw data stored on the hardware, human readable.



16.3 - Attr: Kapooht, CC BY-SA 3.0, via Wikimedia Commons

Understand the operating system as an abstraction level between the end-user and the intricacies of raw data. It provides an interface to the user which hides all the complexities of computer data and how it is being presented.

Before parsing the artifacts and diving deep into analysis, you must fully understand how files are generally handled by an operating system. As mentioned earlier, an operating system is just a very sophisticated piece of software written by the manufacturers to

provide an abstraction level between the complexities of hardware interactions and the user.

In the context of file handling, operating systems either *store* files or *execute* files. Both of which requires different types of memory. Also note that *storing* files requires access to a storage media such as HDDs, SSDs and Flash drives, whereas *executing* files requires access to the microprocessor. Both are handled by the operating system.

As you might already know, computers or any electronic computing device for that matter, primarily utilize two types of memory:

1. **RAM (Random Access Memory)**:
    - Volatile memory, only works for the time power is supplied.
    - Used for assisting execution of applications/software by the processor of the device.

2. **ROM (Read Only Memory)**:
    - Non-volatile memory, retains data even when not in use.
    - Used for storing the application files for a larger period of time.

There are many sub-types of both RAM & ROM but only the fundamental difference between them is concerned here.
Now let's look at the lifecycle of an application in two stages:

1. **Production Cycle**:
An application is a set of *programs*. A program is a set of *code* written by a programmer, generally in higher levelled languages that do not interact directly with machine level entities such as registers, buses, channels etc. That piece of code is written to the disk. The code is then compiled to assembly, which is a lower levelled language which can interact directly with machine level entities. Finally the assembly is converted to the machine code

consisting of 1s and 0s (also known as binary or executable file), which is now ready for it's execution cycle.

2. **Execution Cycle**:
Now that the program is sitting on the disk, waiting to be executed, it is first loaded into the RAM. The operating system instructs the processor about the arrival of this program and allocates the resources when they're made available by the processor. The processor's job is to execute the program one instruction at a time. Now the program can execute successfully if the processor is not required to be assigned another task with a higher priority. If so, the program is sent to the ready queue. The program can also terminate if it fails for some reason. However, finally it is discarded from the RAM.

You can easily remember both of these cycles by drawing an analogy between electronic memory and the human memory. Here, I use chess as an example. Our brains, much like a computer, uses two types of memory:

1. **Short-term (Working memory)**:
    - For a game of chess, we calculate the moves deeply in a vertical manner for a specific line based on the current position.
    - This is calculative in nature. Calculation comes from present situation.

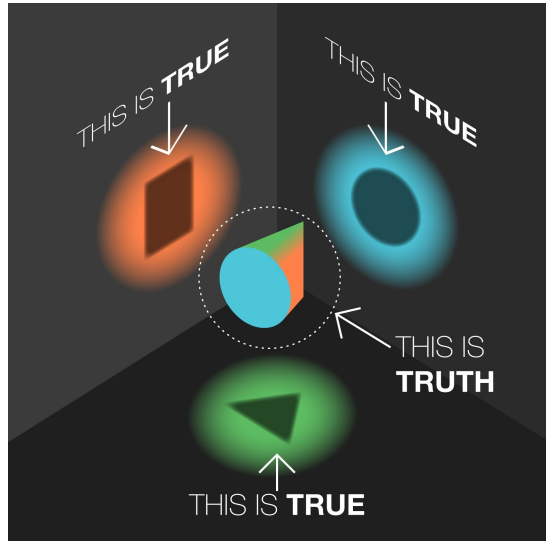
2. **Long-term (Recalling memory)**:
    - At the opening stage in a game of chess, we consider the candidate moves widely in a horizontal manner for many lines.
    - This is instinctive in nature. Instinct comes from past experiences.

Understanding how an operating system parses the data from different sources, rather it be on-disk or in memory; will help you

identify, locate and efficiently retrieve different types of artifacts necessary for an investigation.

Artifact-Evidence Relation

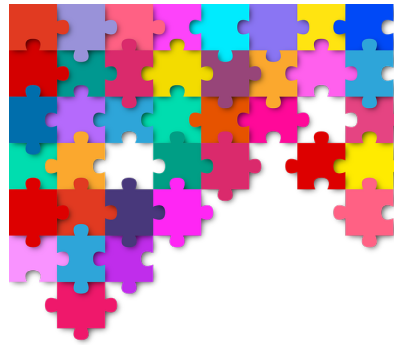
You will come across an ocean of different artifacts in your investigations, but artifacts have a very strange relationship with what might potentially be considered evidence. Artifacts alone do not give you the absolute truth of an event. They provide you tiny peepholes through which you can reconstruct and observe a part of the truth. In fact, one can never be sure if what they have is indeed the truth in it's entirety.



16.4 - Attr: Original by losmilzo on imgur, modified here as text removal

I always love to draw an analogy between the artifacts and the pieces of a puzzle, of which you're not certain to have the edge or

the corner pieces. You gather what you can collect, and try to paint the picture as unbiased and complete as possible.



16.5 - Attr: By stux on pixabay

That being said, if you apply the additional knowledge from metadata, OSINT and HUMINT to the parsed artifacts, you might have something to work with. For instance, say you were assigned an employee policy violation case, where an employee was using their work device for illegally torrenting movies. Parsing the artifacts alone will give you information around the crime, but not as evidence. You would still need to prove that the face behind the keyboard at the time of the crime, was indeed the one that your artifacts claim. So you would then look for CCTV footage around the premises, going back to the ***Identification*** phase in the digital forensics lifecycle, and so forth and so on.

As a result of a codependency of the artifacts on drawing correlations to some external factor, they form a direct non-equivalence relation with evidence. However, note that this "rule", if you will, is only applicaple to a more broad scope of the investigation. In the more narrow scope as a forensicator, and for the scope of your final forensic report, artifacts are most critical. Just keep it in the back of your mind that encountering an artifact alone doesn't mean it's admissible evidence. Parse the artifact, make notes and document everything. Being forensically sound is more important

than worrying about completing the entire puzzle. Because there will be no edge or corner pieces of the puzzle.

Examples

This section will cover how some of the more uncommon artifacts can play into a case from a bird's eye view. We won't be getting into the technical specifics on the parsing or extraction, but the significance of those artifacts at a much higher level. Such as what does it offer, prove and deny. And what is it's forensic value. I suggest the readers to use these brief bits to spark curiosity about these important artifacts, and research on your own about locating and parsing them.

Registry

About:

- Windows registry is a heirarchical database used by the Windows operating system to store it's settings and configurations. Additionally, it also stores some user data pertaining to user applications, activities and other residual traces.
- Registry is structured with what are called Hives or Hive Keys (HK) at the top-most level. Each hive contains numerous keys. A key can contain multiple sub-keys. And sub-keys contain fields with their values.

There are mainly two types of hive files:

1. **System Hive Files**:

- SAM (Security Account Manager): User account information such as hashed passwords, account metadata including last login timestamp, login counts, account creation timestamp, group information etc.
- SYSTEM: File execution times (Evidence of Execution), USB devices connected (Evidence of Removable Media), local timezone, last shutdown time etc.
- SOFTWARE: Information about both user and system software. Operating System information such as version, build, name & install timestamp. Last logged on user, network connections, IP addresses, IO devices etc
- SECURITY: Information about security measures and policies in place for the system.

2. **User Specific Hive Files**:
    - Amcache.hve: Information about application executables (Evidence of Execution), full path, size, last write timestamp, last modification timestamp and SHA-1 hashes.
    - ntuser.dat: Information about autostart applications, searched terms used in windows explorer or internet explorer, recently accessed files, run queries, last execution times of applications etc.
    - UsrClass.dat: Information about user specific shellbags, covered in the next section.

Significance:

- Identifying malware persistence which can lead to the discovery of IOCs.
- Proving the presence of removable media in a particular time frame, which can further help with acquisition of the same.
- Retrieving crackable user password hashes from the SAM and SYSTEM hives, which might help access the encrypted partitions if the password was reused.

Shellbags

About:

- Shellbags were introduced in Windows 7. It is a convenience feature that allows the operating system to remember Windows Explorer configuration for user folders and a folder's tree structure.
- Whenever a folder is created, selected, right-clicked, deleted, copied, renamed or opened, shellbag information will be generated.
- Depending on the Windows version, shellbag information can be stored in either ntuser.dat, UsrClass.dat or both.

Significance:

- Reconstructing the tree structure for deleted folders. Helpful in providing an idea of the files that used to exist when they cannot be carved from the unallocated space.
- Disproving denial of content awareness. If a subject claims that they were simply not aware of something existent on their system, shellbags can disprove their claims with an obvious given that an exclusive usage of the machine was proven.
- Getting partial information about the contents of removable media that were once mounted on the system.

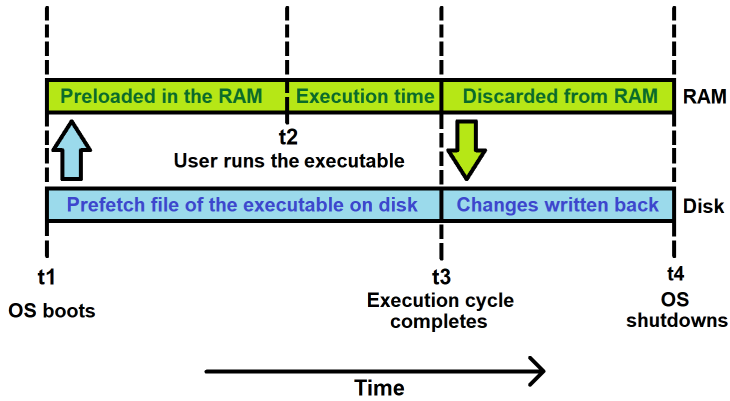
Prefetch

About:

- Prefetch was first introduced in Windows XP. It is a memory management feature which optimizes loading speeds for

files that are frequently executed. Originally it was meant for faster booting times, but since has been developed for applications too. Hence, this artifact is a direct evidence of execution.

- We looked at the lifecycle of an application earlier, the prefetcher in Windows works in the same way. It studies the first ~10 seconds of an application launched and creates/updates the corresponding prefetch file, for faster loading speeds on the next execution.
- Starting from Windows 10, prefetch files are compressed by default to save considerable disk space. It uses the Xpress algorithm with Huffman encoding. For validation purposes, forensicators must decrypt their prefetch files first. Thank you Eric[52], for this handy python script[53] for the same.



16.5 - Working of Windows Prefetcher

It has three working modes:

[52] https://twitter.com/ericrzimmerman
[53] https://gist.github.com/EricZimmerman/95be73f6cd04882e57e6

Value	Description
0	Disabled
1	Application prefetching enabled
2	Boot prefetching enabled
3	Application & Boot prefetching enabled

This value is set from the registry key:
`HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management\PrefetchParameters`
Forensicators can refer to this key to check if prefetching is disabled.

Significance:

- Since it is evidence of execution, it is helpful in identifying anti-forensic attempts at bypassing detection. Any automated anti-forensic tools ran would in turn result in it's own prefetch file being generated.
- Useful in identifying and hunting ransomware executed. Once identified, analysts can look for publicly available decryptors to retrieve encrypted files.
- By studying files and directories referenced by an executable, analysts can identify malware families.
- Application execution from removable media or deleted partitions can be identified from the volume information parsed.
- Timestamps for last eight executions and the run counter is useful for frequency analysis of malicious executables. Worms which would result into multiple prefetch files referencing the exact same resources is a prime example of this.

Jumplists & LNK files

About:

Significance:

SRUDB.dat

About:

Significance:

$MFT

About:

Significance:

$I30

About:

Significance:

$LogFile

About:

Significance:

$UsnJrnl

About:

Significance:

hiberfil.sys

About:

Significance:

References

Chapter 01

Chapter 02

Chapter 03

Chapter 04

Chapter 05

Chapter 06

Chapter 07

Chapter 08

Chapter 09

Chapter 10

Chapter 11

Chapter 12

Chapter 13

Chapter 14

Chapter 15

Chapter 16

16.1 - Scientific law versus Scientific theories[54]

[54] https://creativecommons.org/licenses/by-sa/4.0

16.2 - E-Discovery: An Introduction to Digital Evidence[55]
16.3 - File:Role of an Operating System.svg[56]
16.4 - Our perception of truth depends on our viewpoint 2.0[57]
16.5 - Puzzle Multicoloured Coloured - Free vector graphic on Pixabay[58]

[55]https://www.amazon.com/Discovery-Introduction-Digital-Evidence-DVD/dp/1111310645
[56]https://creativecommons.org/licenses/by-sa/3.0
[57]https://imgur.com/gallery/obWzGjY
[58]https://pixabay.com/vectors/puzzle-multicoloured-coloured-3155663/

Errata

Reporting Errata

If you think you've found an error relating to spelling, grammar, or anything else that's currently holding this book back from being the best it can be, please visit the book's repo on GitHub[59] and create an Issue detailing the error you've found.

- Remove this line prior to final publishing, but finalize the URL once the title is decided upon

[59] https://github.com/Digital-Forensics-Discord-Server/CrowdsourcedDFIRBook/issues

Markdown Example

Writing in Markdown is easy! You can learn most of what you need to know with just a few examples.

To make *italic text* you surround it with single asterisks. To make **bold text** you surround it with double asterisks.

Section One

You can start new sections by starting a line with two # signs and a space, and then typing your section title.

Sub-Section One

You can start new sub-sections by starting a line with three # signs and a space, and then typing your sub-section title.

Including a Chapter in the Sample Book

At the top of this file, you will also see a line at the top that says `{sample: true}`, immediately above the # Chapter Two title. This means that when you generate a preview of your book, or publish a new version of your book, this chapter will be included in a separate sample book that will be created. When you publish your book, this will give potential readers a sample book to read, when they are deciding whether they want to buy your book.

Links

You can add web links easily. Here is a link to the Leanpub homepage[60].

When you are creating a web link, the part between the square brackets [ ] contains the words that people will see in your book, and the part in the round brackets ( ) is the web address you are linking to.

Definition Lists

term 1

: definition 1a

: definition 1b

term 2

: definition 2

Images

You can add an image to your book in a similar way.

First, add the image to the "resources" folder for your book. You will find the "resources" folder in your book's "manuscript" folder, which is at the top level of your book's folder in GitHub or Bitbucket.

If you look in your book's "resources" folder right now, you will see that there is an example image there with the file name "palm-trees.jpg". Here's how you can add this image to your book:

[60] https://leanpub.com

Palm Trees

As this example image shows, **on a line by itself** you type an exclamation point !, followed by the image caption between square brackets and the filename between round brackets.

To learn more about adding images, see this link[61].

Lists

Numbered Lists

You make a numbered list like this:

1. kale
2. carrot
3. ginger

[61] https://leanpub.com/markua/read#images

Bulleted Lists

You make a bulleted list like this:

- kale
- carrot
- ginger

Code Samples

You can add code samples really easily. Code can be in separate files (a “local” resource) or in the manuscript itself (an “inline” resource).

Local Code Samples

Here’s a local code resource:

Hello World in Ruby

```
1  puts "hello world"
```

Inline Code Samples

Inline code samples can either be spans or figures.

A span looks like `puts "hello world"` this.

A figure looks like this:

```
1  puts "hello"
```

You can also add a caption:

Hello World in Ruby

```
puts "hello"
```

To learn more about adding code samples, see this link[62].

Tables

You can insert tables easily inline, using the GitHub Flavored Markdown (GFM) table syntax:

Header 1	Header 2
Content 1	Content 2
Content 3	Content 4 Can be Different Length

To learn more about adding tables, see this link[63].

Math

You can easily insert math equations inline using either spans or figures.

Here's one of the kinematic equations $d = v_i t + \frac{1}{2} a t^2$ inserted as a span inside a sentence.

Here's some math inserted as a figure.

[62] https://leanpub.com/markua/read#code
[63] https://leanpub.com/markua/read#tables

$$\left|\sum_{i=1}^{n} a_i b_i\right| \le \left(\sum_{i=1}^{n} a_i^2\right)^{1/2} \left(\sum_{i=1}^{n} b_i^2\right)^{1/2}$$

Something Involving Sums

To learn more about adding math, see this link[64].

How Book.txt Works

If you look in your book's "manuscript" folder, you will see there is a file called "Book.txt".

The "Book.txt" file is what our book generators use to decide what other files to include in your book. It is a list of the files in your "manuscript" folder that you decide you want to include in your book.

You can have multiple chapters in one file, but we recommend one chapter per file. This way, it's easier to navigate.

If you open the "Book.txt" file now, you will see the following list:

chapter1.txt
chapter2.txt
chapter3.docx
chapter4.docx

If you want to write in plain text, you should delete the following lines in the "Book.txt" file, and save the change:

chapter3.docx
chapter4.docx

(Those files are for people who want to write in Word, not in plain text.)

Now, the only two lines listed in the "Book.txt" file will be:

[64] https://leanpub.com/markua/read#math

chapter1.txt
chapter2.txt

The next time you create generate a preview of your book, our book generators will only use the "chapter1.txt" and the "chapter2.txt" files, because they are the only files listed in the "Book.txt" file. Our book generators will ignore any files in your "manuscript" folder that are not listed in the "Book.txt" file, even if those other files are still in the "manuscript" folder.

Creating a Preview of Your Book

You can generate a new version of your book any time by going to the "Preview" page for your book on Leanpub.

To go to the Preview page for your book, click on the Versions tab on Leanpub when you are working on your book. By default you will be taken to the "Preview New Version" page, which is where you'll be able to preview your book.

Getting Help

Finally, to get help, go to the Help tab for your book in Leanpub and then either…

1. See our Getting Started page to learn how to get started.
2. See our Getting Help page to learn how to get help.
3. Clicking the link to the Markua manual[65] on the Getting Help page. The Markua manual contains everything you need to know about writing in Markdown on Leanpub. (Our dialect of Markdown is called Markua.)

[65] https://leanpub.com/markua/read

By the way, that was how you make a numbered list in Markdown.

You can also make a bulleted list like this:

- item one
- the second item
- another item

We hope that you find writing in Markdown on Leanpub to be simple and enjoyable!