No-Cost Data Exfiltration Artifact Detection

Author: Scott M. Sabo, iamscottsabo@yahoo.com
Advisor: *Bryan Simon*

Accepted: *September 20, 2022*

Abstract

Data exfiltration is an all too often occurrence in businesses all over the world, resulting in not only a monetary loss to the business but other negative effects like reputational damage, market loss, and possibly permanent loss of business. The traditional focus is to keep the bad guys from penetrating the company network and stealing valuable data. But what about insider threats? While enterprise-class tools are available to detect and prevent data exfiltration, smaller businesses may be susceptible to data theft because they cannot afford costly solutions in the same way bigger companies can. This paper looks at two of the most prevalent data exfiltration methods utilized by threats from inside the company. This research will create and forensically examine two scenarios to determine if artifacts can be detected that provide evidence of data exfiltration.

1. Introduction

Data exfiltration is nothing new. While the term "data exfiltration" and "data leak" are often used synonymously, this paper will differentiate between the two. A data leak is when sensitive data is disclosed to an unintended third party (Proofpoint, 2000). This is typically the result of negligence or simply human error. An example of a data leak would be when an employee accidentally e-mails confidential or sensitive data to the wrong recipient. While this is careless and not malicious, it can still cause harm to the originating company.

Data exfiltration, on the other hand, is typically done with purpose. Data exfiltration is defined as "the theft or unauthorized removal or movement of any data from a device" (Fortinet, n.d.). Data exfiltration is theft, pure and simple, and is a serious problem for companies of all sizes. Data theft is the leading type of insider threat activity, with 42% of insider threat events involving the theft of intellectual property or data (Care, 2022). Remote workers are the primary source for insider risk criminal prosecutions. And of those prosecutions, 75% of the investigations that led to those prosecutions involved data theft occurring at home (Care, 2022).

Exfiltration is executed in two ways: via external attacks or insider threats. While both attacks are working toward the same outcome, which is to exfiltrate data from inside the company, they differ in various ways. External attackers must gain access to the organization's systems via an attack vector such as phishing or social engineering. Once the organization has been breached, the attacker works to collect data for exfiltration, most likely using the victim's infrastructure via the Internet as the transport method.

Insider attacks, on the other hand, are typically executed by someone with physical or authorized access to the organization's systems. They are often disgruntled employees looking to inflict harm on the organization for their gain. According to McAfee (2017), internal actors were responsible for 43% of data loss cases. And while the majority of insider data exfiltration uses existing digital communications to steal the data (e-mail or transmission via the Internet), the theft of data via digital media is still quite common (Diana, 2022). Additionally, the use of digital media is more favorable when exfiltration is executed by insiders (McAfee, 2017).

Scott M. Sabo, iamscottsabo@yahoo.com

But why does data exfiltration happen? For cybercriminals, the exfiltrated data is very valuable. Almost any type of data has potential value. From sales and marketing information to customer data to intellectual property, almost any data can be monetized. For example, in 2021, confidential information about Apple's latest iMac products was exfiltrated from a partner company of Apple. The information was held for ransom to the tune of $50 million by the REvil ransomware gang (Bracken, 2021). However, the plan failed when Ukrainian Yaroslav Vasinskyi, the mastermind behind the operation, was arrested and the information was removed from REvil's website (Clover, 2021).

As for insiders, there is a split of 60/40 between employees/contractors and suppliers involved in data theft. According to Pilixo, insiders may steal data for many reasons, including revenge for perceived wrongdoing, a need to meet a financial burden, or just for the thrill of it (Why employees steal data? Insider threat indicators, 2017). Sometimes insiders partner with outsiders. In June of 2021, the LockBit 2.0 ransomware gang began actively recruiting corporate insiders to assist them in ransomware/data exfiltration, promising lucrative payouts (Abrams, 2021). Figure 1 was taken from a ransomware attack screen, soliciting employees to participate in data exfiltration attacks for profit (Abrams, 2021).

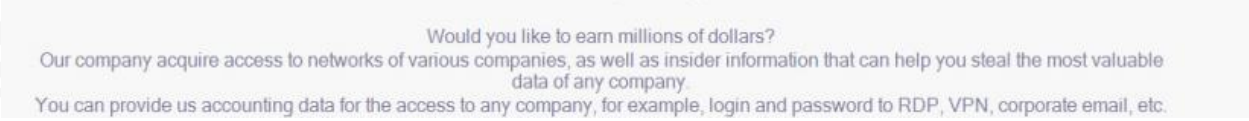


Figure 1 - LockBit 2.0 wallpaper recruiting insiders

1.1. Enterprise Protection

In an organization with a modern Security Operations Center (SOC), one would expect to find various tools to protect the enterprise from data leaks and exfiltration. For example, Intrusion Detection Systems (IDS) would be used to detect and thwart intruders in the network, thus preventing data exfiltration. Likewise, using Data Loss Prevention (DLP) tools would prevent data exfiltration by implementing various monitoring and detection methods, including network traffic inspection or the use of media monitoring tools. However, these tools are outside the scope of this paper.

1.2 Problem Statement

It is possible to discover signs of data exfiltration without using expensive tools. This research will demonstrate how the detection of some of the most common data exfiltration methods can be performed using no-cost tools and methods. Two exfiltration methods will be the primary focus. First is exfiltration over the network. To detect this type of exfiltration an example of network traffic inspection will be given. A demonstration to review network traffic to find evidence of data exfiltrated over the network will be provided. The second method examined is data exfiltration via removable media. Evidence of data exfiltration via removable media will be determined by looking for Universal Serial Bus (USB) artifacts found in the operating system.

2. Research Method

2.1. System Configuration

The testing platform consisted of the following operating systems and tools.

Microsoft Windows 10 Pro
Version 21H2
OS Build 19044.2130
8 Gigabytes RAM

Table 1 - Microsoft Windows Configuration

VMWare Fusion
Version 12.2.4 (2071091)

Table 2 - VMWare Fusion Configuration

Wireshark
Version 4.0.1 (v4.0.1-0-ge9f3970b1527)

Table 3 - Wireshark Version Information

Registry Explorer
V2.0.0.0
https://f001.backblazeb2.com/file/EricZimmermanTools/net6/RegistryExplorer.zip

Table 4 - Registry Explorer Information

LECmd
Version 1.5.0.0
https://github.com/EricZimmerman/LECmd

Table 5 - LECmd Information

REMnux
https://sourceforge.net/projects/remnux/

Table 6 - REMnux Information

2.2. Data set

In real-life data exfiltration cases, data sets can vary from single files to entire massive archives of data. Files can be transmitted as they exist on the source media (without encryption or compression), included in an encrypted archive, or even obfuscated using various techniques. Anything more than plain file transmission makes detection increasingly difficult unless an automated process is used to extract data files from the network transmission stream.

Since the goal is to search for exfiltration detection via network packet inspection as well as USB file evidence, other functions used in data exfiltration detection, such as data obfuscation detection or data transmission volume threshold checking, will not be examined. The data set used for this analysis is very simple, consisting of a plain archive file named “ImportantData.zip” since simple detection methods will be demonstrated.

3. Findings

3.1. Data Exfiltration via Network Communications

To exfiltrate data via a local area network, it is necessary to transmit data across that network. The nature of network communication dictates that the data being sent is split into smaller units called frames or packets. This data can be monitored and captured as it moves across the network on the way to its destination by a process called "sniffing." Sniffing "is a process of monitoring and capturing all data packets passing through a given network" (Grey Campus, n.d.). Sniffing involves one or more devices configured to observe network traffic as it traverses the network. As the traffic passes over the network, sniffers can view the traffic, capture the traffic, or both. Once captured, the traffic can be reassembled using various tools, at which time the contents can be reviewed.

The data can be sent in either one of two ways: unencrypted or encrypted. Unencrypted data packets can easily be captured, reassembled, and viewed using different tools. Reassembly can be performed using an application like Wireshark, the application used in the experiment. Reassembly can also be performed automatically, as is done by network security appliances. When the packet is reassembled, all the network information can be viewed, including the source and destination IP addresses, the protocol, etc. While some information can be gleaned with the network information, what is interesting is the ability to investigate the data stream to determine what was being transmitted. If Wireshark can determine that a file was transmitted, the name and size of the file can be determined. In some cases, the file can be extracted from the reassembled data packets to allow further examination.

Encrypted data, on the other hand, is comprised of network packets that have been transformed from easily read packets to packets where the data payload is encrypted. While the network information is still easily viewed, the data layer is subjected to encryption, transforming, and obfuscation, preventing the viewer from being able to read the contents.

The recommended method for the detection of data exfiltration over the local area network is to observe and inspect the traffic going to an external host on the Internet, most likely using the Hypertext Transfer Protocol Secure (HTTPS – encrypted) HTTPS

uses a security protocol called Transport Layer Security (TLS). TLS secures the communication between the browser and the web host by using an asymmetric public key infrastructure. This process uses two different keys to encrypt the data between two parties:

- Private key: this key is controlled by the owner of the website and is used to decrypt information encrypted by the public key
- Public key: this key is sent by anyone who wants to interact with the server. Data encrypted with the public key can only be decrypted by the private key (Cloudflare, 2022).

The browser and the web server perform a "handshake" process whereby they use the private and public keys to exchange session keys. These session keys are used to securely transmit the data from the web server to the browser and vice versa, up to the point that the communication session is closed by both parties.

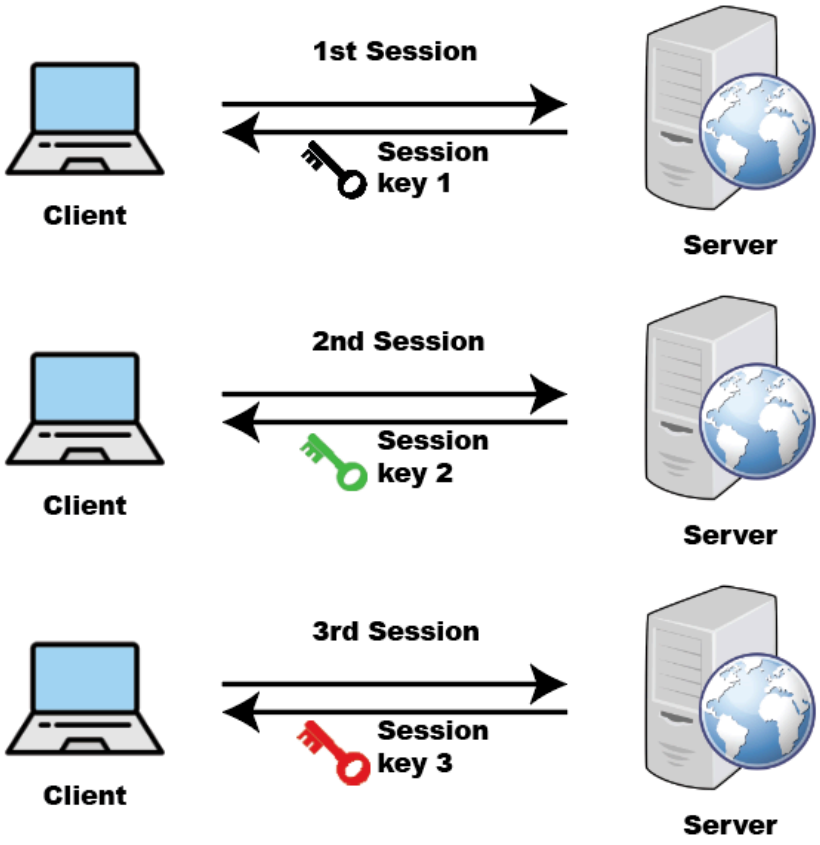


Figure 2- Session Key Exchange

While it is possible to capture the session keys during the session, they are useless unless the encrypted network communication about those session keys is captured. If the analyst has both the encrypted network traffic and the session keys, they can apply the session keys to the encrypted traffic capture and decrypt the network traffic, allowing them to inspect the network traffic, including the formerly encrypted data stream.

3.1.1. Exfiltration of the Data

This experiment will configure a Microsoft Windows system to capture the session keys passed between the browser and the web server. This is done by configuring the Windows system as described in Appendix A. Next, Wireshark will be configured to use the captured session keys to decrypt the encrypted communications. Finally, Wireshark will be used to capture the network communication between Windows and the

web server, as described in Appendix A. The result will be a decrypted session which can be inspected for signs of possible data exfiltration.

To perform the data exfiltration experiment, I created a file named "ImportantData.zip" to use as my test data and selected Microsoft's OneDrive cloud-based storage as my destination. The system the experiment is being run on is a virtual Windows system using Wireshark to perform the packet captures (details of the Windows system and Wireshark can be found in Table 1 and Table 3).

The connection to www.onedrive.com is an HTTPS connection using TLS protection for the connection. This means packets captured by Wireshark will fail to show a plaintext data payload, as shown in Figures 3 and 4.

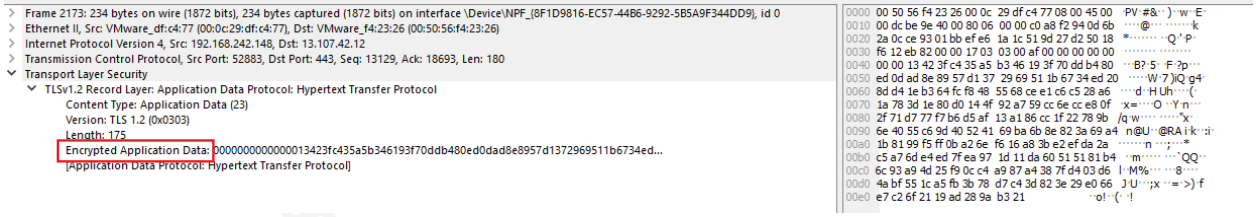


Figure 3 - Encrypted Network Packet

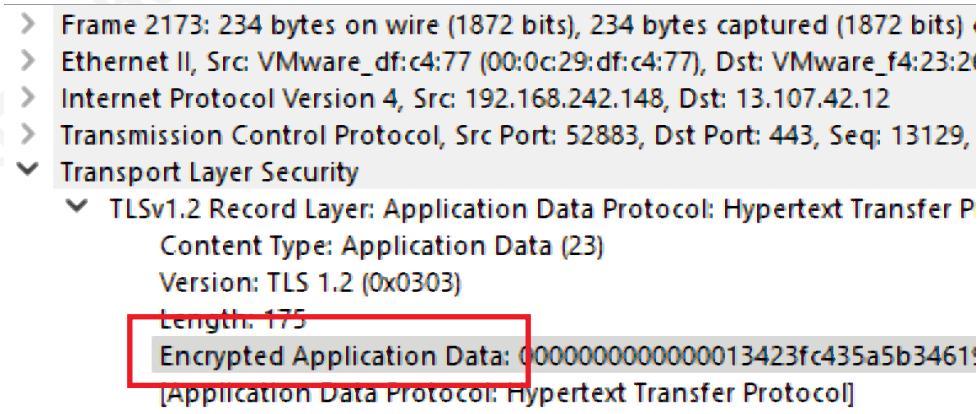


Figure 4 - Detail of Encrypted Network Packet

Since Windows was configured to capture the session keys, Wireshark can be configured to apply those session keys to the network packet capture and decrypt the encrypted traffic! The result is a decrypted data stream, as seen in Figures 5 and 6.

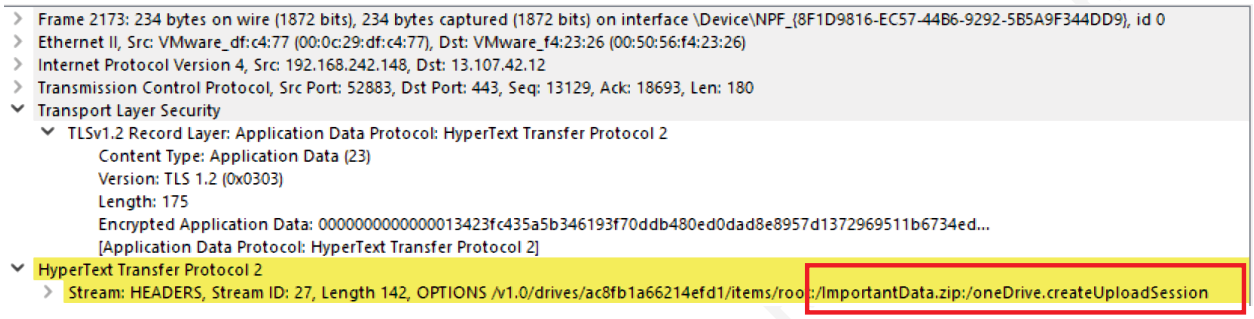


Figure 5 - Decrypted Network Packet

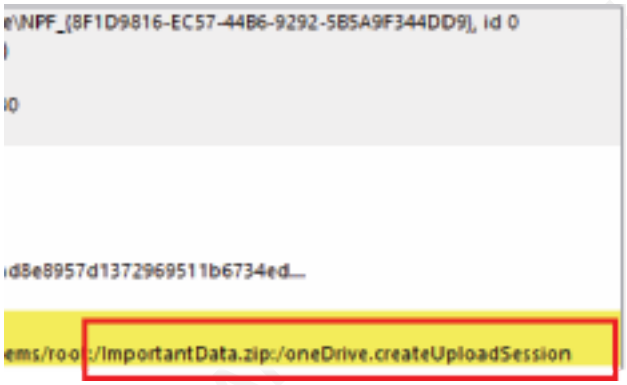
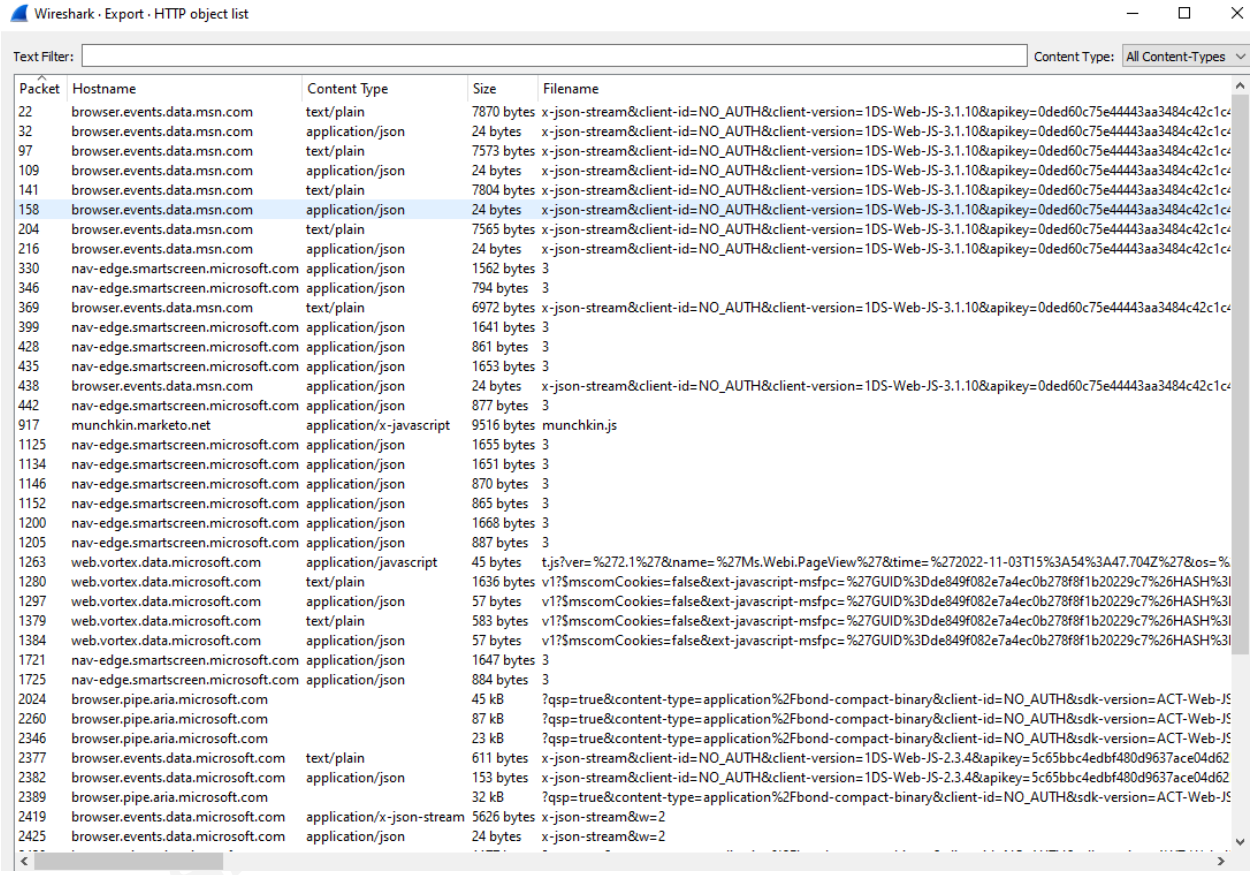


Figure 6- Detail of Encrypted Network Packet

Now that the data stream is decrypted, the details of the data payload can be seen in the packet. Figure 6 shows the contents of the data stream, including the name of the file, "ImportantData.zip."

3.1.2. Difficulties in my Observation

The desired outcome was to be able to extract the file object "ImportantData.zip" from the data stream to allow for further inspection of the file to check for evidence of data exfiltration. Wireshark supports exporting object directly from the packet, but it appeared that Wireshark could not do so successfully. Wireshark would export objects, but none were related to the data upload. This is because Wireshark has difficulty parsing JSON using the current JSON dissector (decoder):



Packet	Hostname	Content Type	Size	Filename
22	browser.events.data.msn.com	text/plain	7870 bytes	x-json-stream&client-id=NO_AUTH&client-version=1DS-Web-JS-3.1.10&apikey=0ded60c75e44443aa3484c42c1c4
32	browser.events.data.msn.com	application/json	24 bytes	x-json-stream&client-id=NO_AUTH&client-version=1DS-Web-JS-3.1.10&apikey=0ded60c75e44443aa3484c42c1c4
97	browser.events.data.msn.com	text/plain	7573 bytes	x-json-stream&client-id=NO_AUTH&client-version=1DS-Web-JS-3.1.10&apikey=0ded60c75e44443aa3484c42c1c4
109	browser.events.data.msn.com	application/json	24 bytes	x-json-stream&client-id=NO_AUTH&client-version=1DS-Web-JS-3.1.10&apikey=0ded60c75e44443aa3484c42c1c4
141	browser.events.data.msn.com	text/plain	7804 bytes	x-json-stream&client-id=NO_AUTH&client-version=1DS-Web-JS-3.1.10&apikey=0ded60c75e44443aa3484c42c1c4
158	browser.events.data.msn.com	application/json	24 bytes	x-json-stream&client-id=NO_AUTH&client-version=1DS-Web-JS-3.1.10&apikey=0ded60c75e44443aa3484c42c1c4
204	browser.events.data.msn.com	text/plain	7565 bytes	x-json-stream&client-id=NO_AUTH&client-version=1DS-Web-JS-3.1.10&apikey=0ded60c75e44443aa3484c42c1c4
216	browser.events.data.msn.com	application/json	24 bytes	x-json-stream&client-id=NO_AUTH&client-version=1DS-Web-JS-3.1.10&apikey=0ded60c75e44443aa3484c42c1c4
330	nav-edge.smartscreen.microsoft.com	application/json	1562 bytes	3
346	nav-edge.smartscreen.microsoft.com	application/json	794 bytes	3
369	browser.events.data.msn.com	text/plain	6972 bytes	x-json-stream&client-id=NO_AUTH&client-version=1DS-Web-JS-3.1.10&apikey=0ded60c75e44443aa3484c42c1c4
399	nav-edge.smartscreen.microsoft.com	application/json	1641 bytes	3
428	nav-edge.smartscreen.microsoft.com	application/json	861 bytes	3
435	nav-edge.smartscreen.microsoft.com	application/json	1653 bytes	3
438	browser.events.data.msn.com	application/json	24 bytes	x-json-stream&client-id=NO_AUTH&client-version=1DS-Web-JS-3.1.10&apikey=0ded60c75e44443aa3484c42c1c4
442	nav-edge.smartscreen.microsoft.com	application/json	877 bytes	3
917	munchkin.marketo.net	application/x-javascript	9516 bytes	munchkin.js
1125	nav-edge.smartscreen.microsoft.com	application/json	1655 bytes	3
1134	nav-edge.smartscreen.microsoft.com	application/json	1651 bytes	3
1146	nav-edge.smartscreen.microsoft.com	application/json	870 bytes	3
1152	nav-edge.smartscreen.microsoft.com	application/json	865 bytes	3
1200	nav-edge.smartscreen.microsoft.com	application/json	1668 bytes	3
1205	nav-edge.smartscreen.microsoft.com	application/json	887 bytes	3
1263	web.vortex.data.microsoft.com	application/javascript	45 bytes	t.js?ver=%272.1%27&name=%27Ms.Webi.PageView%27&time=%272022-11-03T15%3A54%3A47.704Z%27&os=%
1280	web.vortex.data.microsoft.com	text/plain	1636 bytes	v1?$mscomCookies=false&ext-javascript-msfpc=%27GUID%3Dde849f082e7a4ec0b278f8f1b20229c7%26HASH%3
1297	web.vortex.data.microsoft.com	application/json	57 bytes	v1?$mscomCookies=false&ext-javascript-msfpc=%27GUID%3Dde849f082e7a4ec0b278f8f1b20229c7%26HASH%3
1379	web.vortex.data.microsoft.com	text/plain	583 bytes	v1?$mscomCookies=false&ext-javascript-msfpc=%27GUID%3Dde849f082e7a4ec0b278f8f1b20229c7%26HASH%3
1384	web.vortex.data.microsoft.com	application/json	57 bytes	v1?$mscomCookies=false&ext-javascript-msfpc=%27GUID%3Dde849f082e7a4ec0b278f8f1b20229c7%26HASH%3
1721	nav-edge.smartscreen.microsoft.com	application/json	1647 bytes	3
1725	nav-edge.smartscreen.microsoft.com	application/json	884 bytes	3
2024	browser.pipe.aria.microsoft.com		45 kB	?qsp=true&content-type=application%2Fbond-compact-binary&client-id=NO_AUTH&sdk-version=ACT-Web-JS
2260	browser.pipe.aria.microsoft.com		87 kB	?qsp=true&content-type=application%2Fbond-compact-binary&client-id=NO_AUTH&sdk-version=ACT-Web-JS
2346	browser.pipe.aria.microsoft.com		23 kB	?qsp=true&content-type=application%2Fbond-compact-binary&client-id=NO_AUTH&sdk-version=ACT-Web-JS
2377	browser.events.data.microsoft.com	text/plain	611 bytes	x-json-stream&client-id=NO_AUTH&client-version=1DS-Web-JS-2.3.4&apikey=5c65bbc4edbf480d9637ace04d62
2382	browser.events.data.microsoft.com	application/json	153 bytes	x-json-stream&client-id=NO_AUTH&client-version=1DS-Web-JS-2.3.4&apikey=5c65bbc4edbf480d9637ace04d62
2389	browser.pipe.aria.microsoft.com		32 kB	?qsp=true&content-type=application%2Fbond-compact-binary&client-id=NO_AUTH&sdk-version=ACT-Web-JS
2419	browser.events.data.microsoft.com	application/x-json-stream	5626 bytes	x-json-stream&w=2
2425	browser.events.data.microsoft.com	application/json	24 bytes	x-json-stream&w=2

Figure 7 - Wireshark "Export Objects" result

Next, a program named Zeek was used to do the same. Zeek was installed in the REMnux Linux distribution (The link to REMnux can be found in Appendix A). The "extract-all-files" plugin for Zeek was written to export embedded file objects from network packet captures. I verified that the plugin worked by exporting jpg image files from a sample Hypertext Transfer Protocol (HTTP) packet capture which was retrieved from the Wireshark Sample Capture website https://wiki.wireshark.org/uploads/__moin_import__/attachments/SampleCaptures/http_with_jpegs.cap.gz). The command line:

sudo /opt/zeek/bin/zeek -r http_with_jpegs.cap

/opt/zeek/share/zeek/site/packages/file-extraction/plugins/extract-all-files.zeek

successfully exported the image files in the packet capture and saved them to the "extracted_files" folder. The "display" command in REMnux was used to view one of the files for confirmation, as seen in Figure 8.

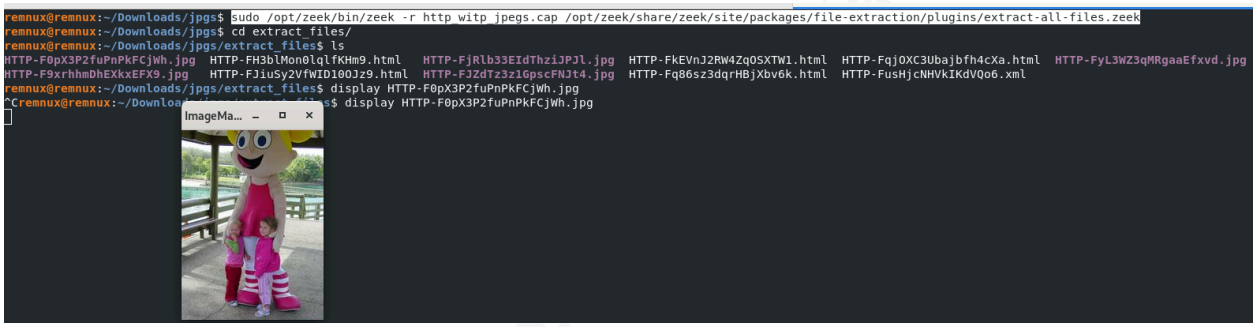


Figure 8 - Successful export of jpg from pcap in Zeek

The same process was performed on the packet capture containing the upload of the "ImportantData.zip" file. However, the only files extracted looked to be the session certificates from the communication between the browser and the web server:

sudo /opt/zeek/bin/zeek -r ImportantData.pcapng /opt/zeek/share/zeek/site/packages/file-extraction/plugins/extract-all-files.zeek

This produced the output shown in Figure 9:

```
remnux@remnux:~/Downloads/caps/ImportantData$ sudo /opt/zeek/bin/zeek -r ImportantData.pcap.pcapng /opt/zeek/share/zeek
/site/packages/file-extraction/plugins/extract-all-files.zeek
remnux@remnux:~/Downloads/caps/ImportantData$ cd extract_files/
remnux@remnux:~/Downloads/caps/ImportantData/extract_files$ ls
SSL-F0DJE44xrsHXBF6qze.ocsp-response     SSL-FDiQys2M5HCbkjwJSe.ocsp-response     SSL-FLpG6y1WDs75sVcWO7.x-x509-user-ce
rt  SSL-FRAAe22ZgMZcrh5kIk.ocsp-response
SSL-F1d1hzSL1Q6SxCBxh.x-x509-ca-cert     SSL-FDNe4FRljd5TPt1Je.x-x509-user-cert   SSL-FMhjto9S95WoJOuL2.x-x509-user-cer
t   SSL-FRJWe03g22BuyGSOO.x-x509-ca-cert
SSL-F3CxZAQSVyKzd1dhe.x-x509-ca-cert     SSL-Fe7U9e45UJknf5xl8c.x-x509-user-cert  SSL-FmQlfc1Jzyu2FByoD7.x-x509-ca-cert
    SSL-FrR9mB4Z3rVzwVYyQf.x-x509-user-cert
SSL-F4Yq2W2dJ4GnwR5GRb.x-x509-user-cert  SSL-FEaMSB2EHkkGOm60va.ocsp-response     SSL-FmttDJ3tC6NTdSojl4.x-x509-user-ce
rt  SSL-FrULRJ2IFbBgm32Zmg.x-x509-ca-cert
SSL-F5ARDh3YimIgPFBh3a.ocsp-response     SSL-FEcFHU24Dz0y3G8gle.x-x509-user-cert  SSL-FMWSjU391wMptxbDWe.ocsp-response
    SSL-FRyL7d1pq29Xb6jj3.x-x509-ca-cert
SSL-F5HHyl2sW94YYvlR4c.x-x509-user-cert  SSL-FeTiC4iJfXsKodnGl.x-x509-user-cert   SSL-FmymCy44nj6NXQuvgi.x-x509-ca-cert
    SSL-FSd3N12AKx38cNyb4f.ocsp-response
SSL-F5niCAlvvDwBUWB9E3.x-x509-ca-cert    SSL-FeuQvK1esohCFF6Io2.x-x509-user-cert  SSL-FnHlJ72FuNSnJdlvTe.x-x509-user-ce
rt  SSL-FSVc0K2KpVkCxFN6A5.ocsp-response
SSL-F618O9I1xr0BtapU8.x-x509-ca-cert     SSL-FeY7RZ3V18tSQ2uL9.x-x509-ca-cert     SSL-FNjCNU1KOhgzrzKFY5.x-x509-ca-cert
    SSL-Fsy8Lt7DMATwoiaH1.ocsp-response
SSL-F65cKp1MXwkKZvep15.x-x509-user-cert  SSL-FfEwGE47TfK0dQTnx1.x-x509-ca-cert    SSL-Fo0ysA3lpLbn1yZgGf.x-x509-ca-cert
    SSL-FtBNR41PEY3d6FbGif.x-x509-ca-cert
SSL-F6QK3v2dCoxvtPnBDk.ocsp-response     SSL-FGkFY44U3FJhe8szGl.ocsp-response     SSL-FoDJIs43MfH7MyWUdj.x-x509-ca-cert
    SSL-FTlYin3lzcFdfPXHk3.x-x509-ca-cert
SSL-F7oSIH36mBOzpOC31d.x-x509-user-cert  SSL-FGkrcT2s3RswyFiiR8.ocsp-response     SSL-FoJLx64x5GqEmwXHPl.x-x509-ca-cert
    SSL-FTzpy95t2DupRchb1.x-x509-ca-cert
SSL-F7R7eYBA09AYlCq5c.x-x509-user-cert   SSL-FgLWaQ3WdQYdOaI0ma.x-x509-ca-cert    SSL-FoKDqU3yVxDC1Vl8B1.x-x509-ca-cert
    SSL-FuB7kI2lHttxBAbdni.x-x509-user-cert
SSL-F88XWw3nM21cnloXN3.x-x509-user-cert  SSL-FgtxG63BxHoesMqdZ1.ocsp-response     SSL-FOrEy6gtS5MX2L1Rh.x-x509-ca-cert
    SSL-FuJLuqlq7JQjXKZh7.x-x509-ca-cert
SSL-F8Te684q76LTfxQPie.x-x509-ca-cert    SSL-Fh16qR2qLdrpgxtQli.ocsp-response     SSL-FPjI6h3pfW2mIVC8Y9.x-x509-user-ce
rt  SSL-FutpCF2QH1t4JSMDbh.x-x509-user-cert
SSL-F8w1IacVl0BhSU9zh.x-x509-user-cert   SSL-FHpNEP3TbS8Lkexct4.ocsp-response     SSL-FpNFRJ3FXw6cX6VtO8.ocsp-response
    SSL-FvBelm2mPuoa25cSyl.x-x509-ca-cert
SSL-FARP7TREgQSJAHph6.x-x509-user-cert   SSL-FIKk7C39386mkwVWR6.x-x509-ca-cert    SSL-FPoCYS1DpuQpTpWKl.x-x509-user-cer
t   SSL-FvDoAh3y9Xc79bz2Ca.x-x509-user-cert
SSL-Faxqxr232FsAykEspb.ocsp-response     SSL-FixUFeU8AqUITYYPk.x-x509-user-cert   SSL-FpOYTN34DJe08E7TN5.ocsp-response
    SSL-FVLfjB261FDXffQoUa.x-x509-ca-cert
SSL-FBjSn3kKQFNCqrho8.x-x509-user-cert   SSL-Fj0nhO3R3HpFamqbPf.x-x509-ca-cert    SSL-Fpzwr212ofdxaGK1Y.ocsp-response
    SSL-FWdwzpFVnMrrZ5Zu8.x-x509-ca-cert
SSL-FbZp1I3NtzbwWzKJ1e.x-x509-ca-cert    SSL-FjagnM2zxzSiwg7xO6.ocsp-response     SSL-FQ8Q5d2ZqOlJAgcj2c.x-x509-user-ce
rt  SSL-FWDyZP1DHK4sAdADs1.x-x509-user-cert
SSL-FciYM54GRxNh5qeRb7.x-x509-ca-cert    SSL-FjWgp81EEJvwD0F8t2.ocsp-response     SSL-FQayPy4I3BMhsHEBo4.x-x509-ca-cert
    SSL-Fwicmt2UePox58sxOc.x-x509-ca-cert
SSL-FCREgk3OZiFq5aVsE1.x-x509-ca-cert    SSL-FjYa424Ojd7A8z7jPl.x-x509-ca-cert    SSL-FQCSFf4WQnOHH3MAtb.x-x509-user-ce
rt  SSL-FwKP6O2Lyb8awJWeH4.ocsp-response
SSL-FcVBXvywb2Ka1RDMh.x-x509-ca-cert     SSL-FL2BX81DQ7JPwmV4Ye.ocsp-response     SSL-FQwkjV21CcqzT5uqq2.x-x509-user-ce
rt  SSL-FzDOhc4tTnFx7YRtT9.x-x509-user-cert
SSL-FDH8uf3An44DLhLOQ3.ocsp-response     SSL-FLGv6ArToURCJOH0l.x-x509-user-cert   SSL-FqX1Ha4VM38fPprA6b.x-x509-user-ce
rt  SSL-FZHopR15KjnHUEM2d6.x-x509-user-cert
remnux@remnux:~/Downloads/caps/ImportantData/extract_files$
```

Figure 9 - Result of Zeek export on packet capture

This was because the session keys were not being passed to Zeek to decrypt the HTTPS sessions. A lot of research was done by looking online for documentation or examples of how to pass the session keys to Zeek, but nothing was found. I appealed to various Zeek groups on the Internet, but I did not receive any responses.

The last attempt to export objects from the packet capture was to use tshark, which is the command line version of Wireshark. With the help of Dr. Johannes Ulrich from the SANS Institute, I was able to determine how to pass the SSL-key.log data (which contained the captured session keys) to tshark to use to decrypt the HTTPS session. Using the command line:

```
tshark -Q -z follow,tls,raw,50 -r Important_Data.pcapng -o tls.keylog_file:ssl-keys.log
```

the file name "ImportantData.zip" was visible, however, the file object was never able to be extracted successfully.

3.2. Removable Media Data Exfiltration

The use of Universal Serial Bus, or USB, devices is a persistent problem in the prevention of insider threat data exfiltration. The media is cheap, the storage capacity is massive, the data is portable, and monitoring employee use of USB devices is not always effective. Even McAfee, the world leader in data loss security, filed a lawsuit in 2019 against three former employees who stole trade secrets via removable USB devices before they left to work for a rival company (Stone, 2019). The focus is to examine activities related to removable USB media to discover if a removable media device was used to steal data.

The primary location to discover USB device use in the Windows operating system is the Windows Registry. The Registry is a collection of databases that hold configuration settings for Microsoft Windows operating systems (Fisher, 2022). Windows stores most of the information and settings for software programs, hardware devices, user profiles and preferences, and operating system configuration in various Registry databases, often referred to as hives (Fisher, 2022). Windows is constantly reading from and writing to the Registry at startup, as the computer is being used as well as during periods of inactivity, and even at the time of shutdown.

The Registry hives contain keys, subkeys, and values, all of which pertain to information relating to the operation of Windows in one way or another (See Table 7).

Hive Name	Purpose
SAM	Stores user's passwords
SECURITY	Stores security policy of user
SOFTWARE	Stores config and settings for applications
SYSTEM	Stores system configuration
NTUSER.DAT	Stores settings and preferences for each user

Table 7- Windows Hives and Their Purpose

When a USB device is connected to a computer running Windows, the operating system records details about the device, such as the vendor, model, and serial number of the device, the date and time when the device was attached, as well as other properties specific to the device. When the device is ejected, Windows also records the date and time of the ejection event.

To perform the investigation, a program called "Registry Explorer" will be used. (The link to Registry Explorer can be found in Appendix A). Registry Explorer allows the analyst to open and browse the Registry files on the suspect computer and gather information related to any USB devices which were attached to the computer. By using Registry Explorer, files can be browsed in a "live" environment, which simplifies the investigation. However, in this "live" environment, Windows locks the Registry files for non-Administrative accounts. This prevents unauthorized access to the Registry to both maintain the integrity of the Registry files as well as to enforce security permissions for non-Administrator accounts. For Registry Editor to have the necessary access to the Registry files it must be run as an Administrator.

3.2.1. Diving Into the Registry

When working through the Registry files, it is helpful to record the information discovered in the "USB Worksheet" which can be found in Appendix C. This keeps track of the data and values as my investigation proceeds. A recommended practice is to record the information on one sheet, as opposed to flipping back and forth between screens. And as a bonus, the completed sheet can be included in the findings of other investigations.

To determine if any USB devices have been used on the suspect computer, first determine which Control Set Windows is currently using. The Control Set is the version of the configuration information Windows uses to perform a successful boot process. There may be more than one Control Set listed in the Registry, depending on any issues Windows may have experienced during previous boot processes. The requirement for accurate USB forensics is to identify the current Control Set, which can be found in the SYSTEM hive.

As shown in Figure 10, navigate to SYSTEM\ROOT\Select to examine the value of the "Current" key. The value is currently set to 1, confirming that the Control Set used

by Windows is ControlSet001.

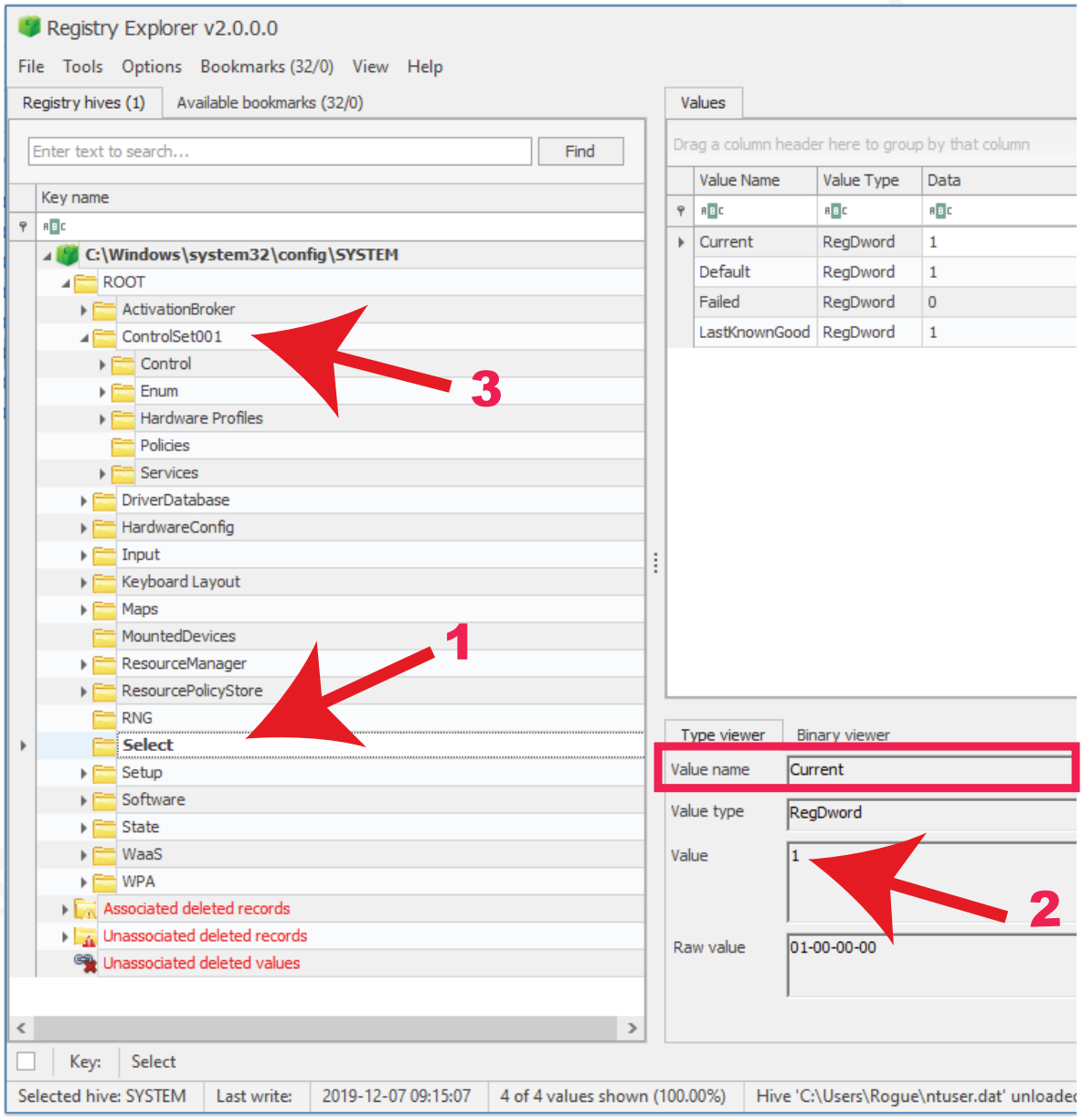


Figure 10- Finding the Current Control Set

Next, navigate to the ControlSet001 key and open the Enum key and the USBSTOR key. The USBSTOR key contains a list of all the USB devices connected to the computer in the last 30 days. As shown in Figure 11, only two devices are listed, however, they should both be investigated to determine if they were used for data

exfiltration.

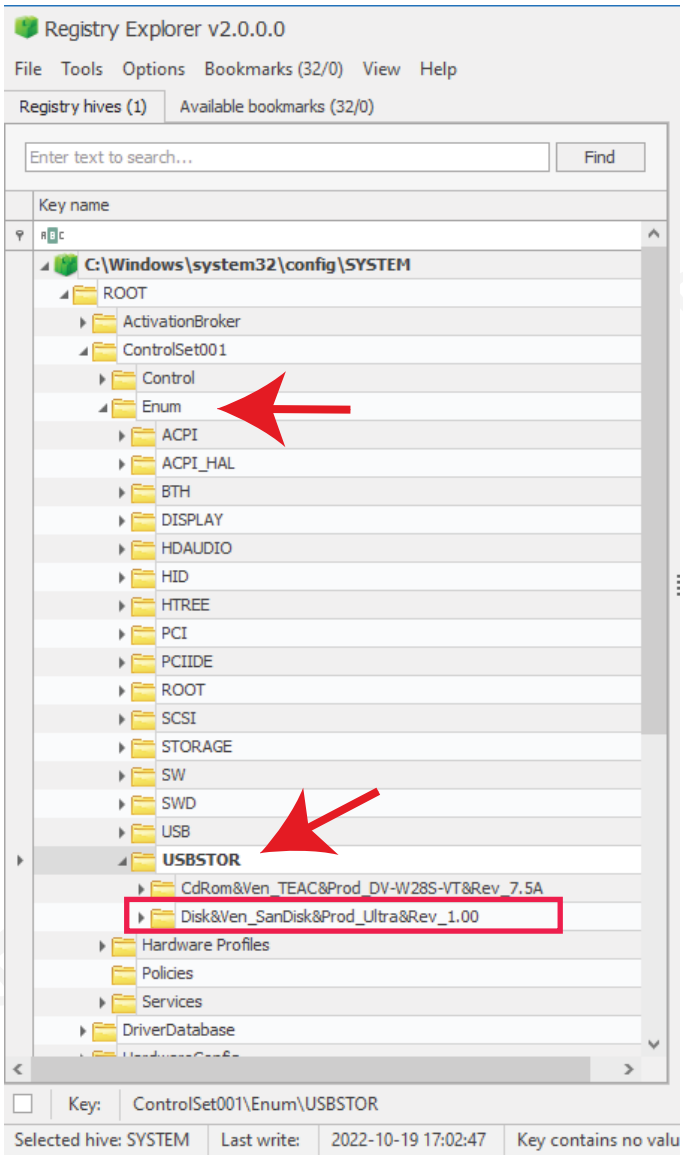


Figure 11- List of USB Devices

The top device (CdRom&Ven_TEAC&Prod_DV-W28S-VT&Rev_7.5A) is a CD-ROM device and cannot be used to write data. The bottom device (Disk&Ven_SanDisk&Prod_Ultra&Rev_1.00) looks to be worth investigating based on the string “SanDisk” observed in the device name.

For each device, there will be a unique label that contains the vendor’s name as well as the product name and a version number. In the example above, the following can be seen:

Disk&Ven_SanDisk&Prod_Ultra&Rev_1.00	
Vendor	SanDisk
Product	Ultra
Version	1.0

Table 8 - Vendor Name and Product

The next value to find is the device serial number. This can be found by opening the device key. As you can see in Figure 12, Registry Explorer shows that there are two serial numbers listed for the same device name. This indicates that two different SanDisk USB devices of the same type were plugged into the computer.

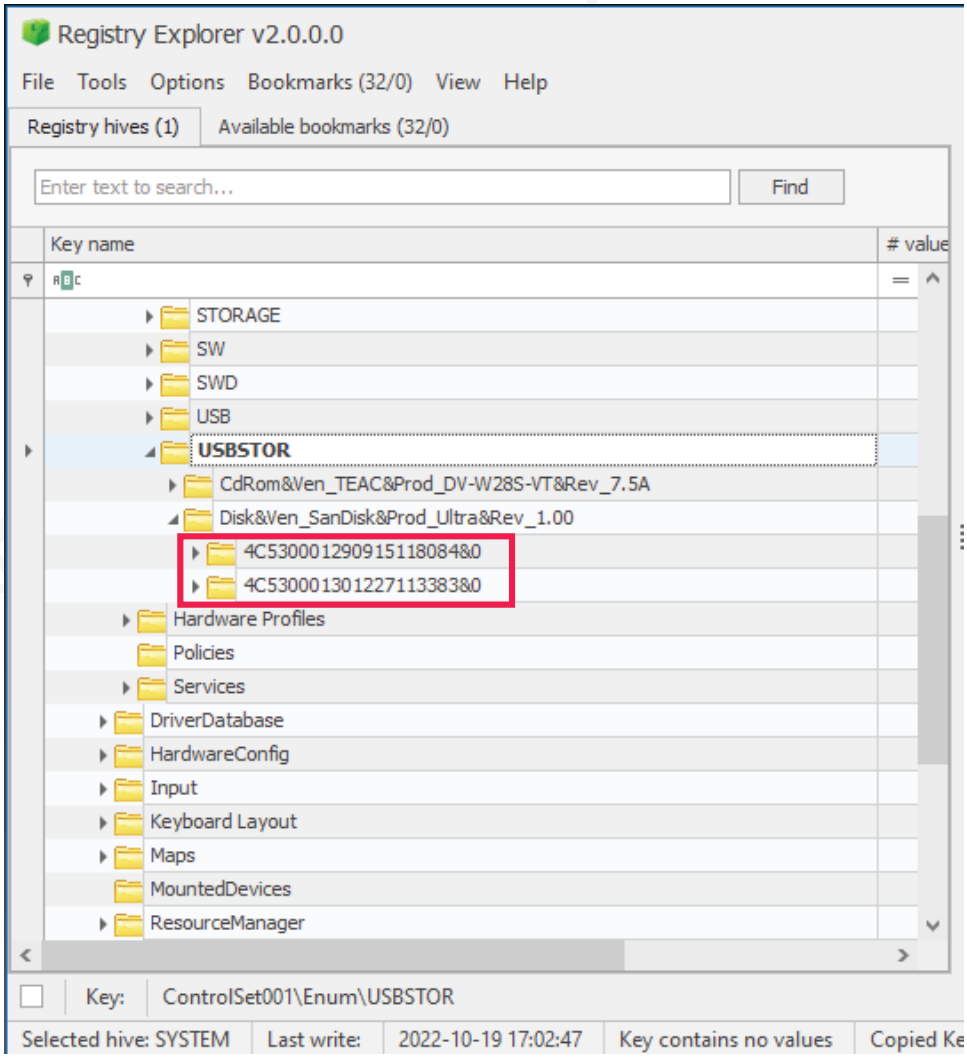


Figure 12 - Serial Numbers Indicating Two Different SanDisk USB Devices

Next, the Vendor ID (VID) and Product ID (PID) for the device or devices (if there are more than one listed) need to be discovered. This is done by navigating to the SYSTEM\USB and opening all the keys with the prefix VID and looking for the serial numbers recorded on the worksheet. In this case, the VID is 0781 and the PID is 5581, as seen in Figure 13.

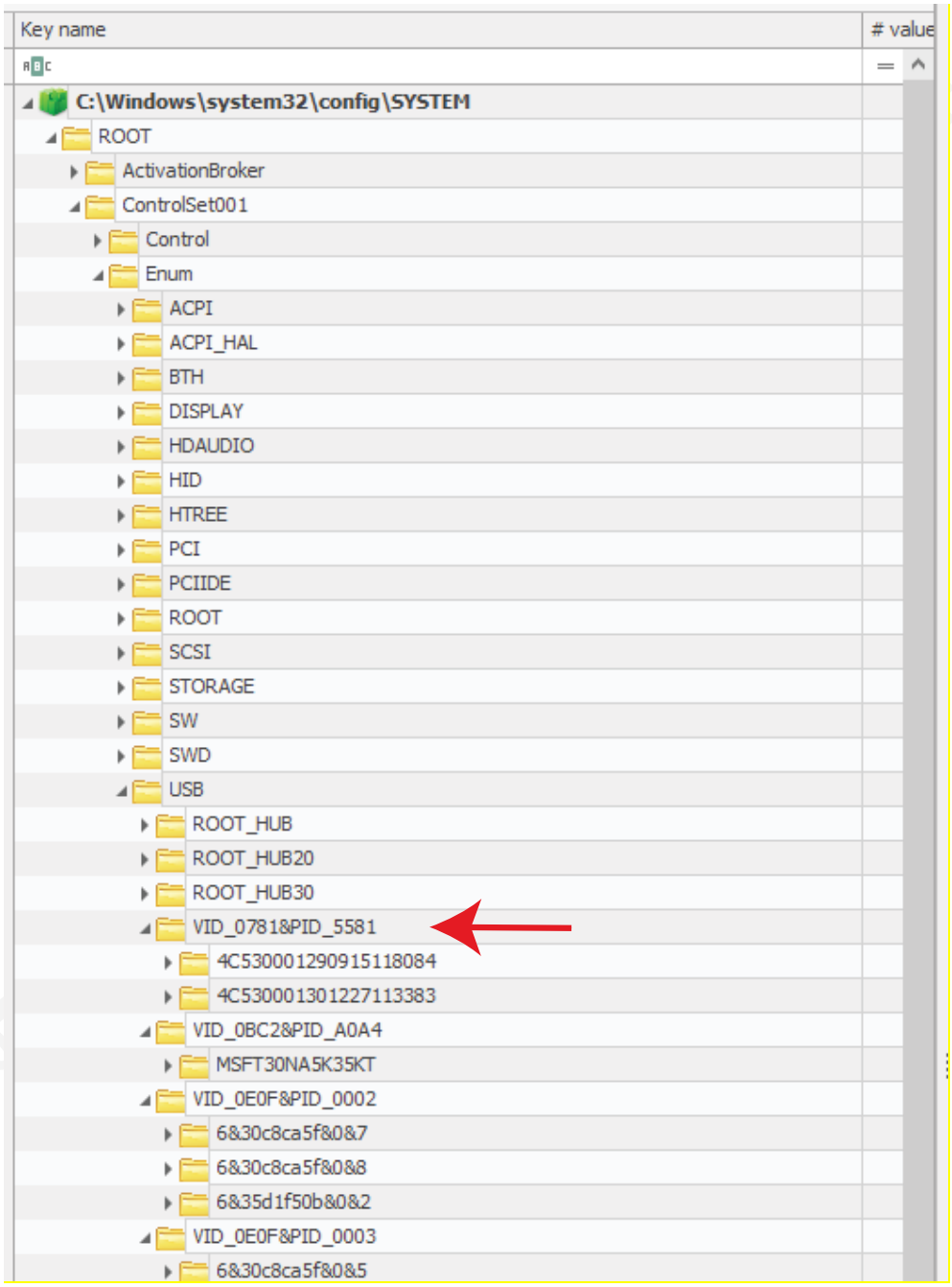


Figure 13 - Finding the VID and the PID

The Volume Name, which is the next critical piece of data, can be found in the SOFTWARE hive in the following location:

SOFTWARE\ROOT\Microsoft\Windows Portable Devices\Devices

It is important to note that there may be more than one device with the same name, as is the case for the target computer. If there are multiple devices, both volume names should be recorded in the worksheet.

Notice below that there are two devices with the same name. After selecting the device in the left column, the Volume Name will be displayed on the Values pane. In this case, the Volume Name is “Data2Go”:

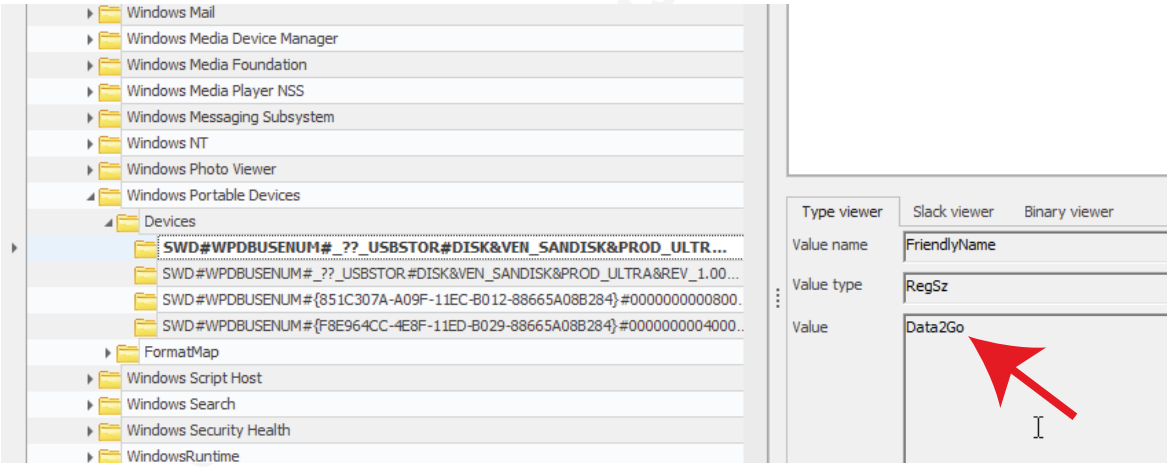


Figure 14- Volume Name

The drive letter assigned to the USB device after it was mounted by Windows can be found by navigating to:

SYSTEM\ROOT\MountedDevices

Figure 15 shows the list of Mounted Devices which will be found on the right side of Registry Explorer. The “Values” tab is selected at the top and the serial number of my device is searched in the “Type Viewer” tab. It was discovered that the serial number of the device is associated with the E: drive letter.

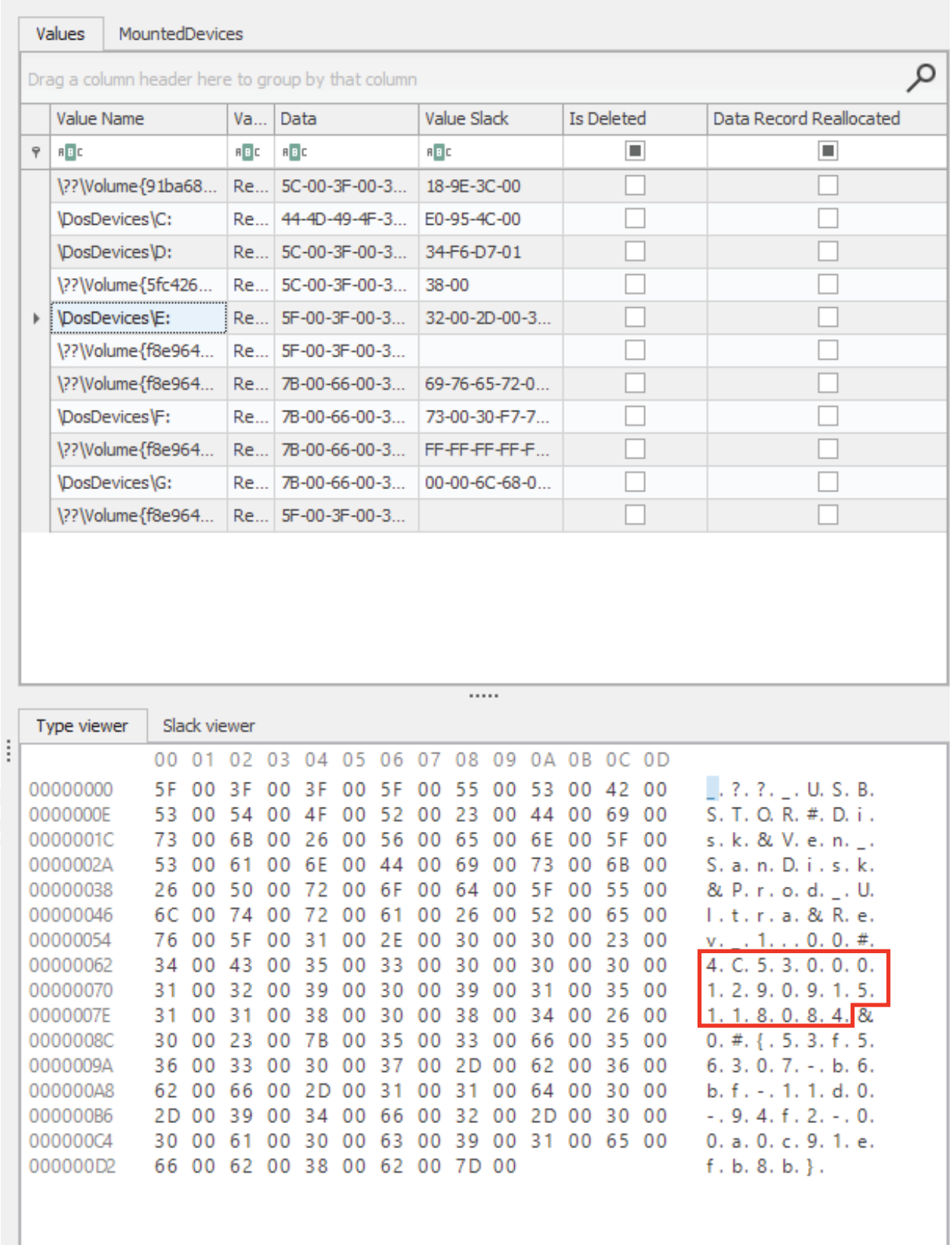


Figure 15 - Associating the Serial Number with the Drive Letter

Next, the Volume GUID needs to be found and recorded. This can be found by clicking on the long strings located in the "Values" tab which are not related to a drive letter and looking for the device serial number in the "Type Viewer" area. Once the serial number is found, record the Volume GUID. In this case, the Volume GUID is {f8e964f3-4e8f-11ed-b029-88665a08b284}.

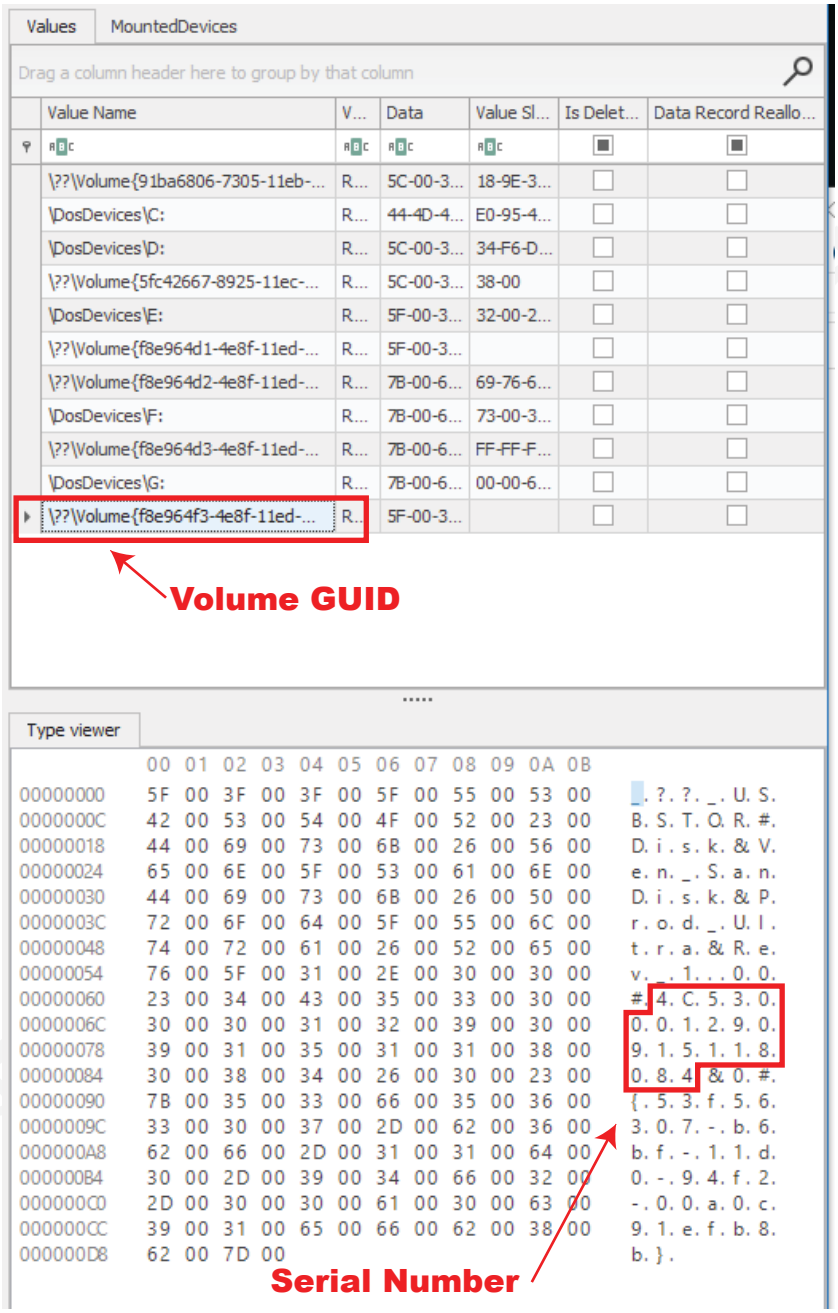


Figure 16- Determining the Volume GUID based on the Serial Number

Now that most of the forensic details of the USB device used on the target computer have been assembled, it is important to identify the user who was logged in at the time the USB drive was attached to the computer. To do so, examine the NTUSER.DAT files found on the computer.

Windows creates an NTUSER.DAT file for each user on the computer. The NTUSER.DAT can be found in the root of the user's profile directory (ex: C:\Users\Rogue\ntuser.dat) and contains specific information about the user account settings and preferences for the specific user. For this system, two NTUSER.DAT files are related to the two user accounts on this system: Rogue and iamsc, as shown in Figure 17:

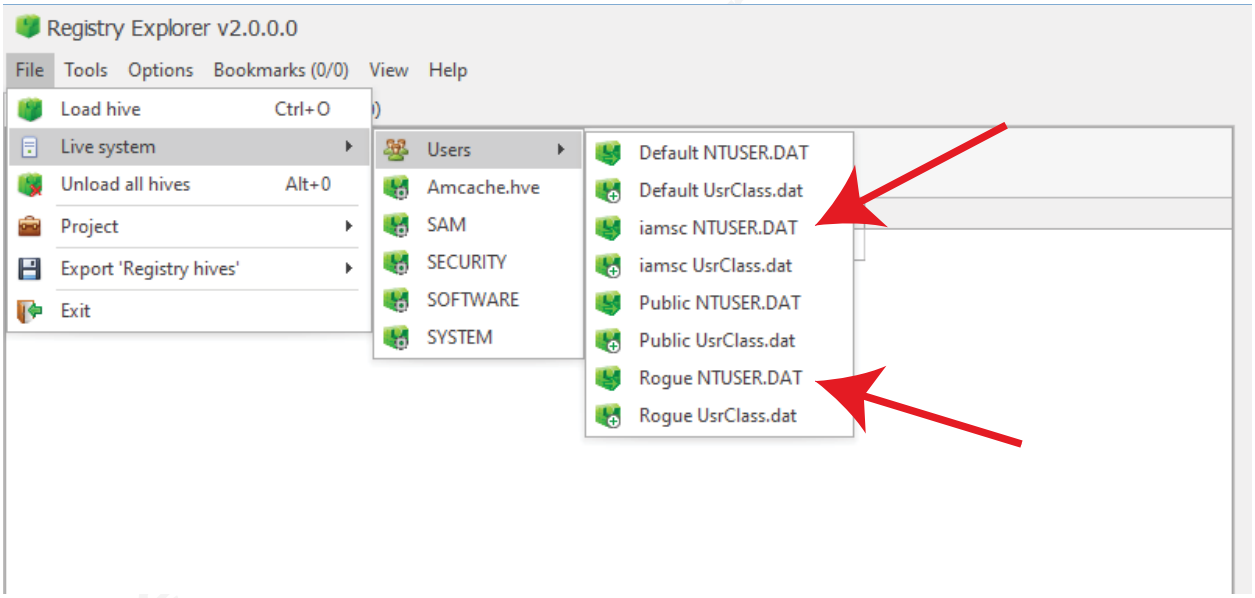


Figure 17 - Illustration of NTUSER.DAT Files for User Profiles iamsc and Rogue

In Registry Explorer, the NTUSER.DAT file for each user is loaded and the following key is examined:

NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2

If any of the GUID values found in SYSTEM\MountedDevices key are found in the listing of the GUIDs in the NTUSER.DAT, the USB drive can be attributed to the owner of the NTUSER.DAT. However, to be sure that only one user used the USB device, all the available NTUSER.DAT files need to be checked for the existence of the GUID mentioned above.

Upon examining the NTUSER.DAT file for the user Rogue, the Volume GUID "{f8e964f3-4e8f-11ed-b029-88665a08b284}" can be observed, which attributes the USB to the user Rogue.

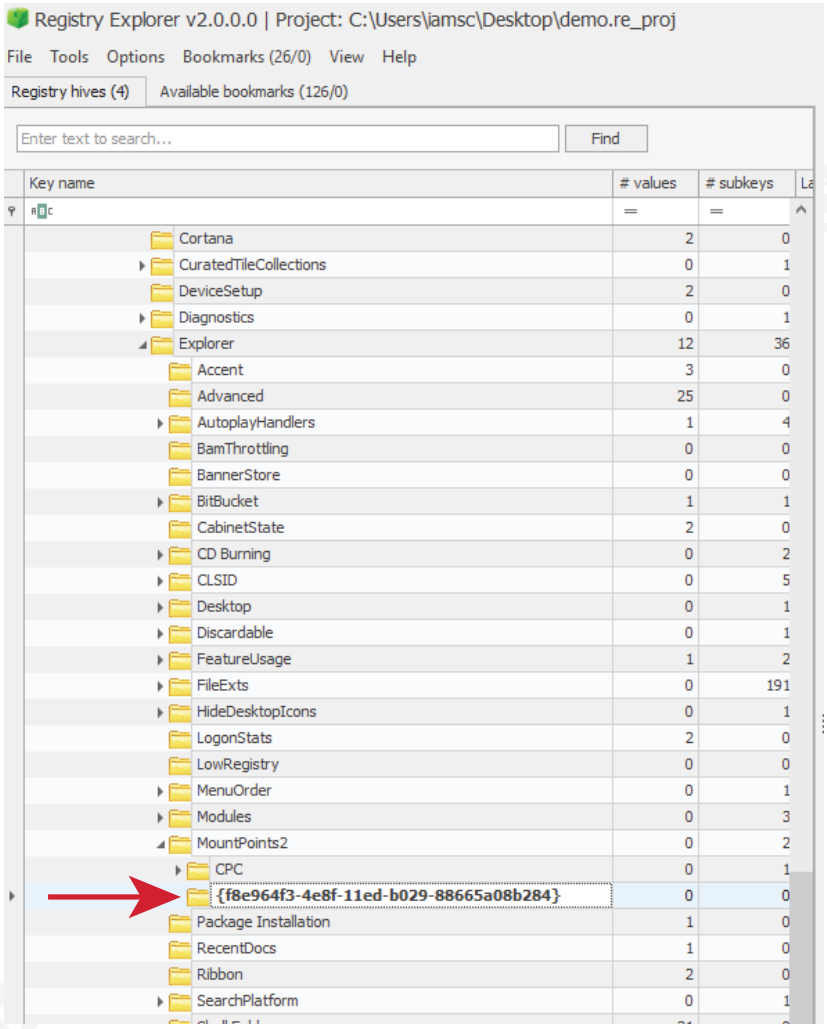


Figure 18 - GUID attributed to UserID Rogue

The first time the USB device was connected to the computer can be discovered by looking at the following key:

SYSTEM\ControlSet001\Enum\USBSTOR\DeviceName\Properties\ 83da6326-97a6-4088-9453-a1923f573b29\

Note that the device name varies by device, but the GUID “83da6326-97a6-4088-9453-a1923f573b29” is not unique to the computer being examined. It can be found on any Windows computer examined.

The following sub-keys for the three date and time stamps can be observed under the GUID “83da6326-97a6-4088-9453-a1923f573b29” as shown in Figure 19:

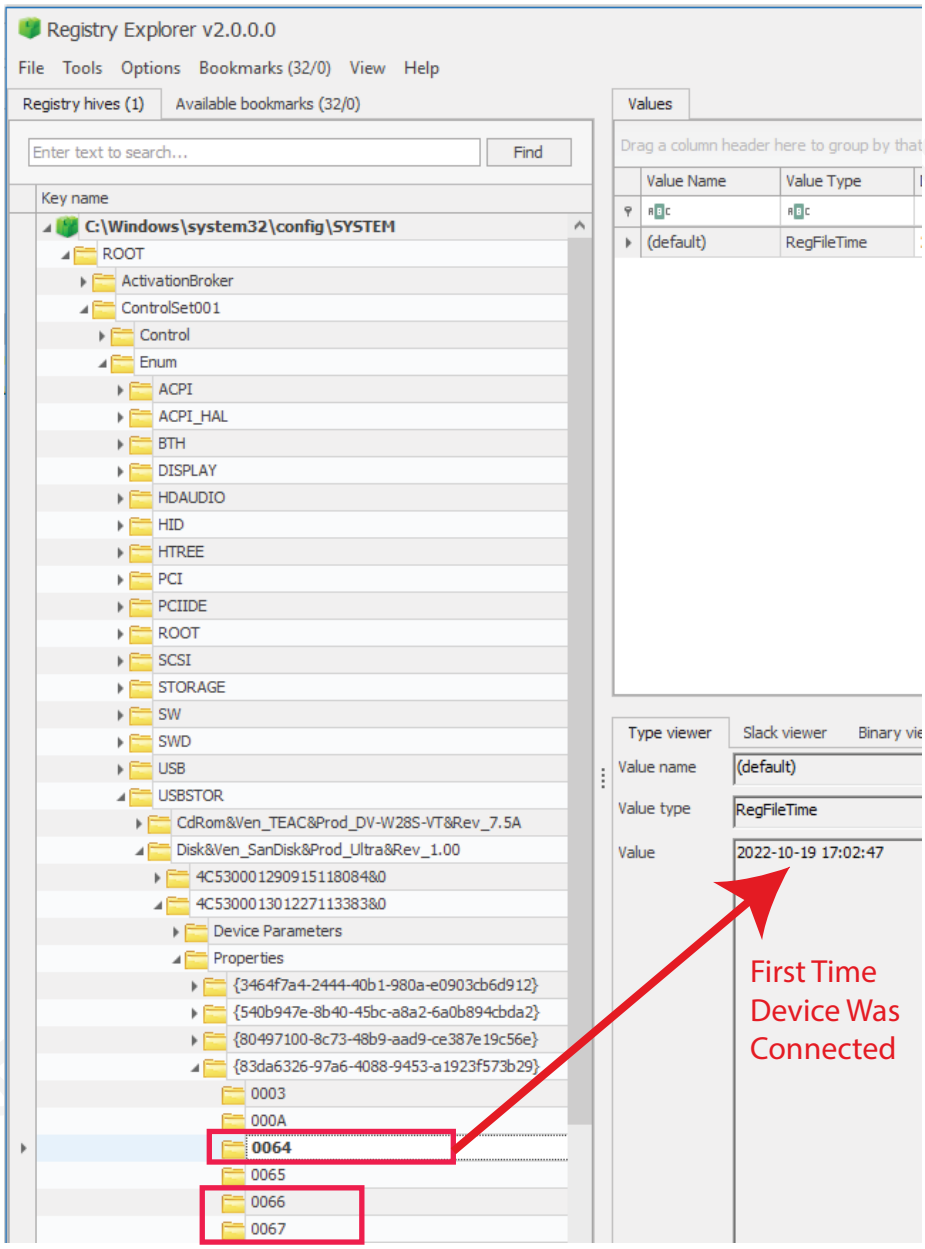


Figure 19 - Location of USB Device Time Stamps and Detail of First Time Device Was Connected

Property	Subkey	My Device Value
First time device was connected	0064	2022-10-19 17:03:32
Last time device was connected	0066	2022-11-02 01:09:07
Last time device was disconnected	0067	2022-11-04 01:52:13

Table 9 - Property, Subkey, and Value of USB Activity Timestamps

Scott M. Sabo, iamscottsabo@yahoo.com

The dates and times the USB device was first connected to the PC, the last time it was connected to the PC, and the last time it was disconnected from the PC are now all known. With the amount of information collected, analysts can move forward in the investigation.

3.2.2. Further Exfiltration Evidence

In an enterprise environment, analysts can use the data gathered by agents installed on the endpoint (computer), providing a step-by-step view of the events that happened on the computer. Other times devices are processed in a "dead-box" forensic mode, where forensic analysts capture an image of a storage device and analyze the contents offline (thecyberwire.com, n.d.).

I expected to find a method to be able to examine the Windows operating system in its "live state," meaning that it is powered up and running, to find some evidence of the copy of my test file "ImportantData.zip" to the USB storage device.

The research shows that unless a file is opened on the destination USB media (with an appropriate application), there isn't any evidence at all. However, if an application were used to open a file on the USB drive, Windows would create a Link (LNK) file associated with that specific action. An LNK file is a shortcut file that is generated when a user opens a local or remote file.

There are some key artifacts associated with the LNK file:

- The original path of the file
- The modify, access, and change (MAC) times of the original file
- Information about the volume and system where the LNK file is stored
- Network details (if the file opened was stored on a network drive)
- The file size of the linked file

3.2.3. Link File Analysis

I wanted to look for further confirmation, so I added a situation into my scenario where the user Rogue wanted to make sure the file "ImportantData.zip" would open on the USB drive. Rogue did this by navigating to the USB drive while it was plugged into the computer and double-clicking on the file to open it with an archiving tool called 7zip.

Being satisfied with what they saw, Rogue ejected the USB drive and took the file "ImportantData.zip" right out the door.

While the contents of the file which was copied to the USB drive cannot be verified, the LNK file can be scrutinized to see what evidence can be found. To do this, a program written by Eric Zimmerman called "LECmd" was used (the link to LECmd can be found in Appendix A). LECmd is a utility used to parse LNK files, meaning that it will examine and display the data found in the LNK file.

LNK files are found in the following path using Windows Explorer:
C:\Users\%USERNAME%\AppData\Roaming\Microsoft\Windows\Recent Items
or the following path if you're using the command prompt:
C:\Users\%USERNAME%\AppData\Roaming\Microsoft\Windows\Recent
The command window will be used to run the LECmd utility.

When a directory listing is performed in the directory which contains the LNK files, C:\Users\Rogue\AppData\Roaming\Microsoft\Windows\Recent, a link file for the "ImportantData.zip" file called "ImportantData.lnk" can be observed:

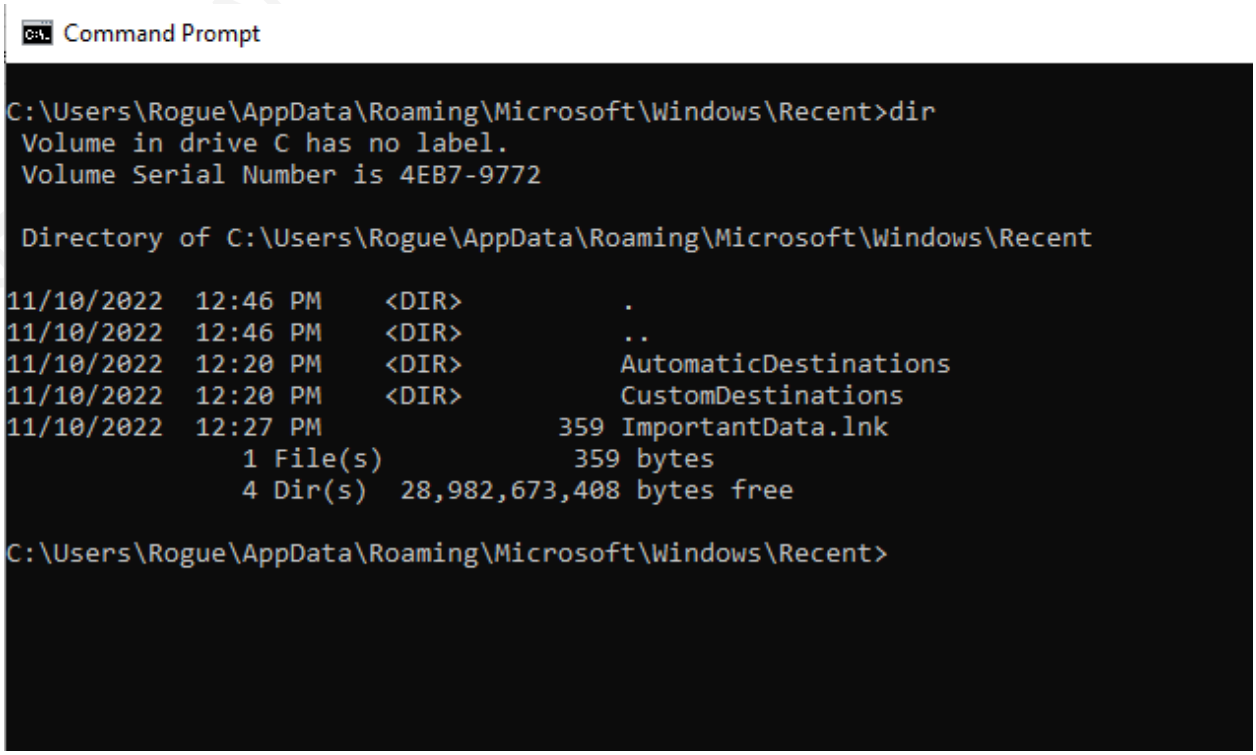


Figure 20- Location of ImportantData.lnk

When the LECmd command is run against the "ImportantData.lnk" file using the command line LECmd.exe -f ImportantData.lnk
the following output is shown in Figure 21:

```
C:\Users\Rogue\AppData\Roaming\Microsoft\Windows\Recent>c:\users\Rogue\Documents\Tools\LECmd\LECmd.exe -f ImportantData.lnk
LECmd version 1.5.0.0

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

Command line: -f ImportantData.lnk

Warning: Administrator privileges not found!


Processing C:\Users\Rogue\AppData\Roaming\Microsoft\Windows\Recent\ImportantData.lnk

Source file: C:\Users\Rogue\AppData\Roaming\Microsoft\Windows\Recent\ImportantData.lnk
  Source created:  2022-11-10 20:27:52
  Source modified: 2022-11-10 20:27:52
  Source accessed: 2022-11-10 20:56:18

--- Header ---
  Target created:  2022-11-01 13:54:32
  Target modified: 2021-11-22 19:35:48
  Target accessed: 2022-11-01 13:54:32

  File size: 147,810
  Flags: HasTargetIdList, HasLinkInfo, IsUnicode, DisableKnownFolderTracking
  File attributes: FileAttributeArchive
  Icon index: 0
  Show window: SwNormal (Activates and displays the window. The window is restored to its original size and position if the window
is minimized or maximized.)


--- Link information ---
Flags: VolumeIdAndLocalBasePath

> Volume information
  Drive type: Removable storage media (Floppy, USB)
  Serial number: 3AEDDA37
  Label: Data2Go
  Local path: E:\ImportantData.zip

--- Target ID information (Format: Type ==> Value) ---

  Absolute path: E:\\

  -Users property view: Drive letter ==> E:\

  -File ==> (None)
    Short name: ImportantData.zip
    Modified:    2021-11-22 19:35:48
    Extension block count: 1

    --------- Block 0 (Beef0004) ---------
    Long name:
    Created:     2022-11-01 13:54:34
    Last access: 2022-11-01 13:54:32
    MFT entry/sequence #: 5415296/null (0x52A180/0xnull)

--- End Target ID information ---

---------- Processed C:\Users\Rogue\AppData\Roaming\Microsoft\Windows\Recent\ImportantData.lnk in 0.11295780 seconds ----------
```

1

2

3

Figure 21 - Output of LECmd command

The data in Detail 1 provides the time stamp when the LNK file was created. This is the time when the source of the LNK file, ImportantData.zip, was opened. Detail 2 provides

the volume label of the USB drive (Data2Go) and the associated path to the file, opened (E:\ImportantData.zip). And finally, Detail 3 shows the Absolute Path to the file associated with the LNK file, which is E:\. **Note: The serial number shown in Detail 2 is the volume serial number assigned by the operating system at the time the drive is formatted and not the device serial number (as noted in Figure 12).**

The Powershell cmdlet "Get-Acl" is used to determine the LNK file's owner. Powershell is run from the Command Prompt and then "Get-Acl" is against the LNK file with the Powershell command line

Get-Acl C:\Users\Rogue\AppData\Roaming\Microsoft\Windows\Recent\ImportantData.lnk

This produces the output shown in Figure 22, showing that Rogue is the owner of the LNK file.

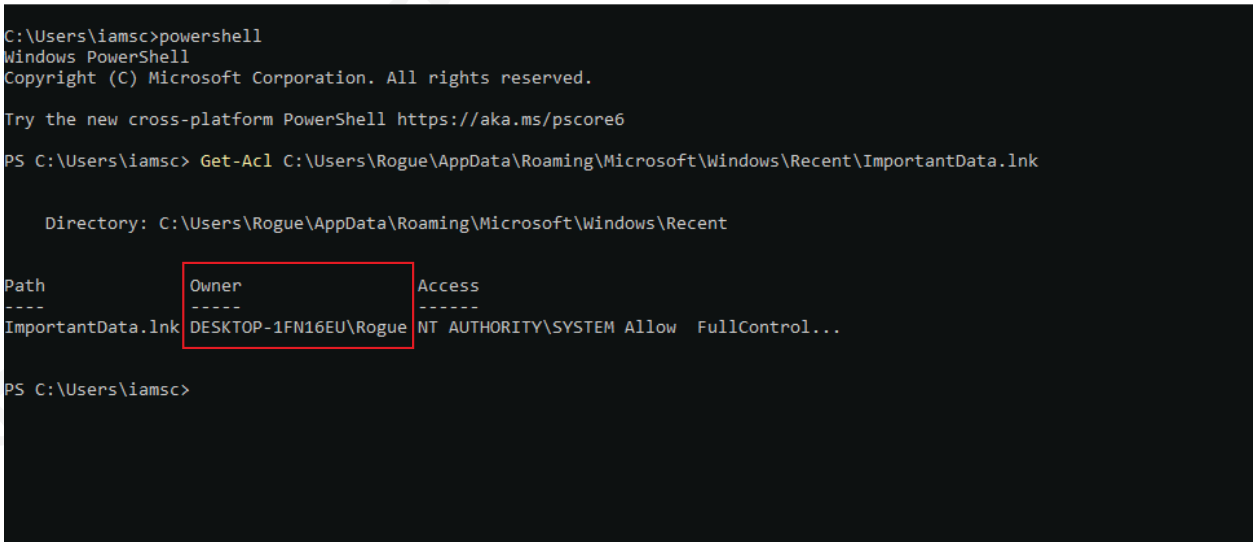


Figure 22 - Evidence of Rogue Owning the LNK file

The evidence correlates well with the evidence found in my earlier investigation of the Registry hives via Registry Explorer and can be used to move forward in the investigation into the data theft event. I have been able to match the Volume Name, the assigned drive letter, the creation of the LNK file under Rogue's directory space, and ownership of the LNK file by the user Rogue on the target computer.

4. Recommendations and Implications

Data exfiltration is a real threat to companies. Implementing no-cost solutions like the ones demonstrated in this paper can provide relative information to the analyst, but are time-consuming, technically awkward, and require many steps to produce the data needed for analysis. Furthermore, the analysis performed in my experiments was not done in real time, which is impractical if the goal is to detect and prevent data exfiltration.

4.1. Recommendations for Practice

Both of my experiments produced results offering evidence of data exfiltration. However, the results were lacking in completeness regarding the level of detail required to draw an accurate conclusion surrounding the events. In the case of data exfiltrated over the network, it would be necessary to be actively looking at and capturing network communication traffic at the very moment the sensitive data was being sent to the external web server. For data exfiltration performed via removable media, while it was possible to discover a substantial amount of data surrounding the use of the removable USB media, it was not possible to identify what was being copied to the media.

It is recommended to use tools that provide continuous monitoring of the two paths of data exfiltration I identified. This would allow for continuous inspection, detection, and alerting for exfiltration over the network as well as via removable USB media.

4.2. Implications for Future Research

The primary limitation for my experiments revolved around understanding how to get any of the tools (Wireshark, tshark, or Zeek) to properly parse the packet capture, allowing me to extract the file object. A lack of experience in the tools needed to dissect various protocols in this area resulted in a lot of time being spent researching and learning about the tools.

I reached out to others for assistance. I posted questions to online communities including Git, Reddit, and Stack Overflow, for assistance, but I did not get any responses. This was disappointing. Likewise, I reached out to various professionals in the

Information Security profession. I did get one response which helped me to get a partial answer. I then took this information and attempted to get further on the problem, but to no avail.

As for the other tools, everything ran well and was well documented. To preserve my Windows and REMnux virtual machine environments, I took frequent snapshots in VMWare Fusion, just in case. I did have some issues with Registry Explorer not loading hives. In this case, I simply had to close the application and re-open it.

In the future, it would be helpful to better understand how to completely decrypt encrypted network traffic to facilitate the inspection of the data stream. This would allow the analyst to extract any file objects for further inspection to aid in the detection of data exfiltration.

5. Conclusion

The approach chosen for this research paper was to create and forensically analyze two realistic, real-world data exfiltration methods, one of exfiltration to an external website and the other to exfiltration via removable media. These two methods were chosen because they are two of the more likely methods of exfiltration (McAfee, 2017), and they are most likely to leave evidence on the local computer.

While there are many data loss prevention (DLP) tools available on the market which are made to prevent data theft, these experiments/this research focused on forensic methods and tools which avoid using enterprise-class tools. This allows anyone to consider this research and implement the process which works or avoid the processes that don't work. Microsoft Windows was selected as the operating system as the test platform as Microsoft Windows is the current market share leader with 88% of the market share (Blumer, 2022). Finally, I used widely available tools (Wireshark / tshark, Zeek, LECmd, Registry Explorer, and Powershell) as the investigative tools.

The first experiment regarding data theft via the network was difficult. Having a deeper understanding of network traffic analysis would have made the experiment easier. I was able to apply the TLS session keys against the packet capture which allowed me to view the name of the file being exfiltrated. This could be considered suspicious,

prompting a deeper investigation. However, for the file extraction portion of the experiment, my experiment did not completely achieve the results I was hoping for. I solicited assistance from others knowledgeable in the topic, and while I was able to make some progress, file extraction never worked correctly.

As for the experiment exploring data exfiltration via removable USB, a significant amount of data was discovered which could be used to support an investigation. Registry Explorer provided key details about the USB devices used including the device names, the serial numbers, and other important details like when the devices were used. LECmd provided a correlation of the data found by Registry Explorer by examining the LNK file to find the Volume Label and the drive letter assigned to the device, as well as the timestamps related to when Rogue opened the suspicious file. Finally, the LNK file was attributed to the user Rogue by running the Powershell cmd-let "Get-Acl." All this information can be used as part of the investigation into possible data exfiltration.

References

13CUBED. (n.d.). DFIR Cheat Sheet. Retrieved November 9, 2022, from https://www.13cubed.com/downloads/dfir_cheat_sheet.pdf.

Abrams, L. (2021, August 6). Lockbit ransomware recruiting insiders to breach corporate networks. BleepingComputer. Retrieved November 6, 2022, from https://www.bleepingcomputer.com/news/security/lockbit-ransomware-recruiting-insiders-to-breach-corporate-networks/

BlackFog. (2022, October 5). What is data exfiltration and how can you prevent it? Retrieved November 6, 2022, from https://www.blackfog.com/what-is-data-exfiltration-and-how-can-you-prevent-it/#What_are_the_long-term_costs_of_data_exfiltration

Blumer, Andrew. (2022, November 8). Microsoft Market Share for 2022. Retrieved November 10, 2022, from https://comparebrokers.co/compare/microsoft-market-share/

Bracken, B. (2021, April 23). Revil's Big Apple Ransomware gambit looks to pay off. Threatpost.com. Retrieved November 6, 2022, from https://threatpost.com/revil-apple-ransomware-pay-off/165570/

Care, J., Furtado, P., & Predovich, B. (2022, April 18). Market Guide for Insider Risk Management Solutions. Gartner. Retrieved November 6, 2022, from https://www.gartner.com/doc/reprints?id=1-29RPXPBT&ct=220418&st=sb

Cloudflare. (n.d.). What is a session key? Session keys and TLS handshakes. Retrieved November 7, 2022, from https://www.cloudflare.com/learning/ssl/what-is-a-session-key/

Cloudflare. (n.d.). What is HTTPS?. Retrieved November 8, 2022, from https://www.cloudflare.com/learning/ssl/what-is-https/

Clover, J. (2021, November 8). DOJ arrests Hacker involved with Revil group that stole Apple's MacBook Pro Schematics. MacRumors. Retrieved November 9, 2022, from https://www.macrumors.com/2021/11/08/revil-group-hacker-arrest/

Deyarmond, S. (2022, May 26). Forensic Analysis of LNK files. Retrieved November 10, 2022, from https://www.magnetforensics.com/blog/forensic-analysis-of-lnk-files/

Diana. (2022, October 14). Data exfiltration: Most common techniques and best prevention tactics. Retrieved November 7, 2022, from https://www.xorlab.com/en/blog/data-exfiltration-most-common-techniques-and-best-prevention-tactics

Fisher, T. (2022, April 19). What is the windows registry? Retrieved November 7, 2022, from https://www.lifewire.com/windows-registry-2625992

Fortinet. (n.d.). Data Exfiltration. Retrieved November 11, 2022, from https://www.fortinet.com/resources/cyberglossary/data-exfiltration

Grey Campus. (n.d.). Sniffing and its Types. Retrieved November 10, 2022, from https://www.greycampus.com/opencampus/ethical-hacking/sniffing-and-its-types

McAfee. (2017). Grand Theft Data. Retrieved November 6, 2022, from https://www.mcafee.com/enterprise/en-us/assets/reports/rp-data-exfiltration.pdf

Pilixo. (2017, September 1). Insider Threat Indicators: Why Employees Steal Data. Retrieved November 9, 2022, from https://www.pilixo.com/insider-threat-indicators/index.htm

Proofpoint. (2022, July 27). What is a data leak? - definition, Types & Prevention. Retrieved November 6, 2022, from https://www.proofpoint.com/us/threat-reference/data-leak

Rath, L. (2022, August 24). How to decrypt SSL with Wireshark in 2022. ITT Systems. Retrieved November 8, 2022, from https://www.ittsystems.com/decrypt-ssl-with-wireshark/#wbounce-modal

Stone, J. (2019, June 25). Former McAfee employees conspired to take 'secret sauce' to Tanium, lawsuit says. Retrieved November 21, 2022, from https://www.cyberscoop.com/mcafee-lawsuit-tanium-employees-secret-sauce/

Appendix A

To enable Wireshark to decrypt network traffic encrypted via TLS, the workstation needs to be configured to capture the session keys generated by HTTPS communications and you must use either the Firefox or Chrome browser.

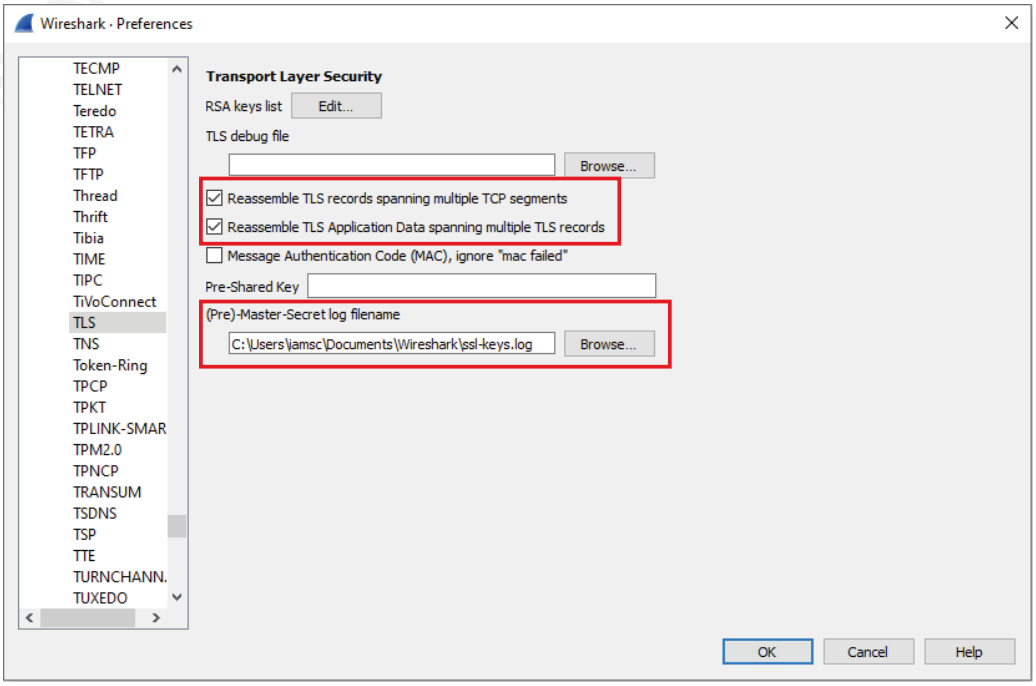
1. Create an environment variable
    - Go to “Settings” in Windows and click on “Advanced Settings”
    - Select the “Environment Variables” button
    - Create a new environment variable called SSLKEYLOGFILE (this will automatically be populated by Firefox or Chrome)
    - Select a location where you would like to store the log file. I stored mine in C:\Users\iamsc\Documents\Wireshark\ssl-keys.log
    - Click OK and close the Windows settings box
    - It may be necessary to restart Windows for the environment variable to be implemented
    - Open the Firefox or Chrome Internet browser and navigate to any website which uses HTTPS for its transmission protocol
    - Check the log file you specified in SSLKEYLOGFILE to make sure it is being populated with session key information

Appendix B

Configure Wireshark to use Session Keys

You must configure the operating system to be able to capture session keys first. The process to configure Windows to capture session keys can be found in Appendix A.

1. Download and Install Wireshark
2. Configure Wireshark
   - Open Wireshark and navigate to Edit >Preferences
   - Select “Protocols” and then scroll down and select “TLS”
   - In the Transport Layer Security options, enter the path to the log file created by SSLKEYLOGFILE (see Appendix A) in the dialogue box under “(Pre)-Master-Secret log filename”
   - Make sure “Reassemble TLS records spanning multiple TCP segments” and “Reassemble TLS Application Data spanning multiple TLS records” are both checked



   - Select “OK”

Once the steps above are completed you should be able to capture and decrypt TLS traffic.

Scott M. Sabo, iamscottsabo@yahoo.com

Appendix C

USB Worksheet

Vendor	SYSTEM\CurrentControlSet\Enum\USBSTOR
Value	

Product	SYSTEM\CurrentControlSet\Enum\USBSTOR
Value	

Version	SYSTEM\CurrentControlSet\Enum\USBSTOR
Value	

Serial Number	SYSTEM\CurrentControlSet\Enum\USBSTOR
Value	

VID	SYSTEM\CurrentControlSet\Enum\USB
Value	

PID	SYSTEM\CurrentControlSet\Enum\USBSTOR
Value	

Volume	SOFTWARE\Microsoft\Windows Portable Devices\Devices
Value	

Drive Letter	SYSTEM\Mounted Devices
Value	

Volume GUID	SYSTEM\Mounted Devices
Value	

User	NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2
Value	

Volume GUID	SYSTEM\Mounted Devices
Value	

Volume Serial Number	SYSTEM\Mounted Devices
Value	

First Time Connected	SYSTEM\CurrentControlSet\Enum\USBSTOR\Ven_Prod_Version\USB_Serial#\Properties\{83da6326-97a6-4088-9453-a1923f573b29}\0064
Value	

Last Time Connected	SYSTEM\CurrentControlSet\Enum\USBSTOR\Ven_Prod_Version\USB_Serial#\Properties\{83da6326-97a6-4088-9453-a1923f573b29}\0064
Value	

Last Time Disconnected	SYSTEM\CurrentControlSet\Enum\USBSTOR\Ven_Prod_Version\USB_Serial#\Properties\{83da6326-97a6-4088-9453-a1923f573b29}\0064
Value	