Lab #2: Malware Analysis

Inor Wang

Department of Cyber Security, The University of Texas at San Antonio

IS 3523.004

Professor Munoz

March 6, 2025

Table of Contents

Introduction

This lab provides a detailed examination of the core responsibilities faced by cybersecurity professionals and incident responders, using a legacy Windows XP environment

that contains multiple unpatched vulnerabilities and outdated software. Although modern organizations typically rely on current operating systems, older machines are still prevalent in industrial applications, smaller businesses, and specialized hardware. By conducting a thorough investigation, from scanning for open ports and collecting memory images to analyzing suspicious executables and scrutinizing system logs, participants gain valuable insight into the attack vectors that malicious actors often exploit, as well as the forensic artifacts that can reveal such activity. Emphasis is placed on maintaining the integrity of digital evidence through proper chain-of-custody procedures and on documenting each investigative step, ensuring both replicability and legal admissibility of findings. Ultimately, by tackling the challenges inherent to this legacy operating system, students cultivate a foundational set of forensic skills and incident response techniques that remain relevant across diverse and evolving technological landscapes.

Objectives

The primary objective of this lab is to conduct a full-scale forensic investigation of a potentially compromised Windows XP system, employing a comprehensive set of analysis and evidence-gathering techniques. Participants use tools such as Memoryze, Redline, and Autopsy to capture volatile memory, isolate hidden processes, analyze registry artifacts, and uncover malware signatures or suspicious executables. In addition to these core activities, the lab places significant emphasis on examining network services, such as open FTP ports, remote shells, and backdoor connections (for instance, via Netcat), to illuminate how attackers initially gained access and maintained persistence. By correlating discovered artifacts, ranging from event logs, Prefetch data, and registry entries to anomalous network traffic, with the system's operational context (including installed applications and known user accounts), investigators can piece together a clear timeline of the compromise. This approach addresses each phase of a potential attack, from the method of initial infiltration and subsequent privilege escalation to the establishment of persistence and any possible data exfiltration methods used by the intruder. Throughout the process, the lab underscores best practices for forensic evidence handling, including maintaining a proper chain of custody, compressing and transferring large memory images without corrupting their integrity, and systematically documenting all investigative steps. Such meticulous recordkeeping not only facilitates peer review but also supports the reproducibility of findings and the potential legal admissibility of evidence. Ultimately, the lab aims to illustrate how older, unpatched systems remain prime targets for unauthorized activity and to highlight the essential role of thorough forensic workflows. By integrating memory, filesystem, and network analysis, students gain the expertise required to detect, contain, and remediate breaches, skills that remain vital across both legacy and contemporary computing environments.

Accessing the Compromised VM

To begin the process of investigating or exploiting a target machine, it is crucial to verify that you can actually reach the host over the network. In this lab environment, I first logged into SimSpace and found an available Windows XP machine labeled "win-xp-xx," specifically "win-xp-15." I navigated to the Network tab of "win-xp-15" and took note of the IP address listed

under the ISCS-Security Subnet connection, as shown in Figure 1. This subnet-specific IP is essential because it is the address I must use to connect to and interact with the Windows XP system. An incorrect IP would prevent any remote access attempts because we would be either trying to access an invalid IP address or the wrong machine.

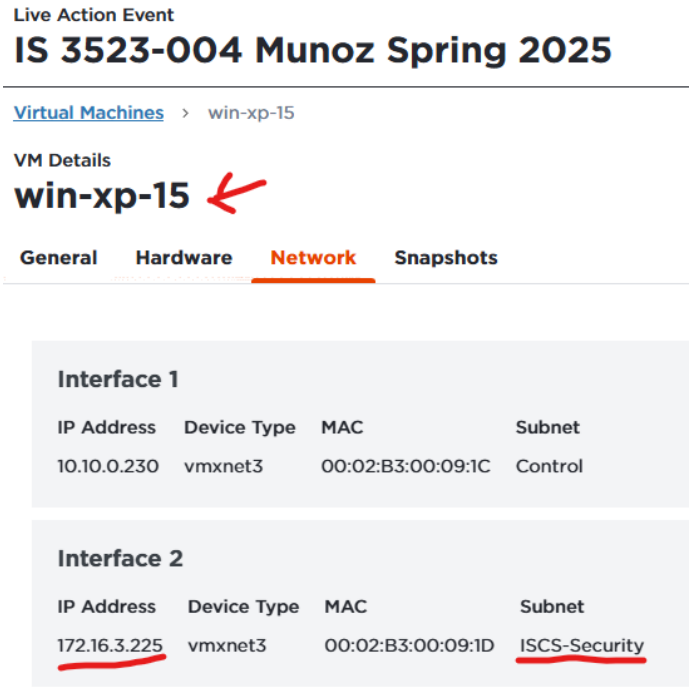

Figure 1: IP address of "win-xp-15" on the ISCS-Security Subnet connection

Once I had the IP address, I launched the console for the Windows XP virtual machine to get a sense of its current state. I observed the default login screen as shown in Figure 2, which showed a user named “Daniel Faraday” but, naturally, I do not have this user’s password. Since I cannot log in directly, my strategy is to access the machine remotely from a Windows 10 system, designated as “win-hunt-18,” to carry out any necessary steps for the lab. Before attempting FTP or any other service access, it is standard best practice to ping the target machine’s IP address from the host you’ll be working on. In this case, I opened a terminal on “win-hunt-18” and issued a simple ping command to the “win-xp-15” IP as shown in Figure 3. Receiving a successful reply to all four ping packets as shown in Figure 3, meant that I could proceed confidently with additional tasks on the Windows XP machine. Since the network path was confirmed to be working, the next step was to move on to the actual exploitation process such as trying to access the FTP server, gathering system information, or attempting to reset passwords.

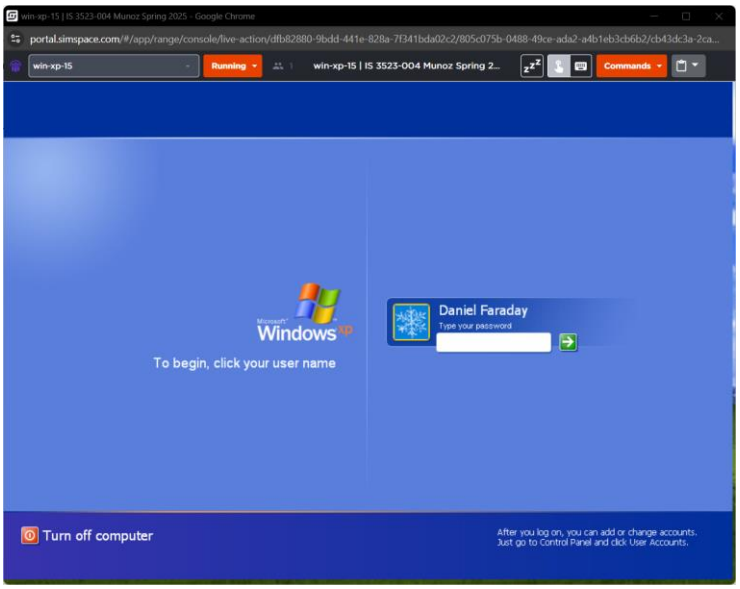

Figure 2: Initial Windows XP log-in screen

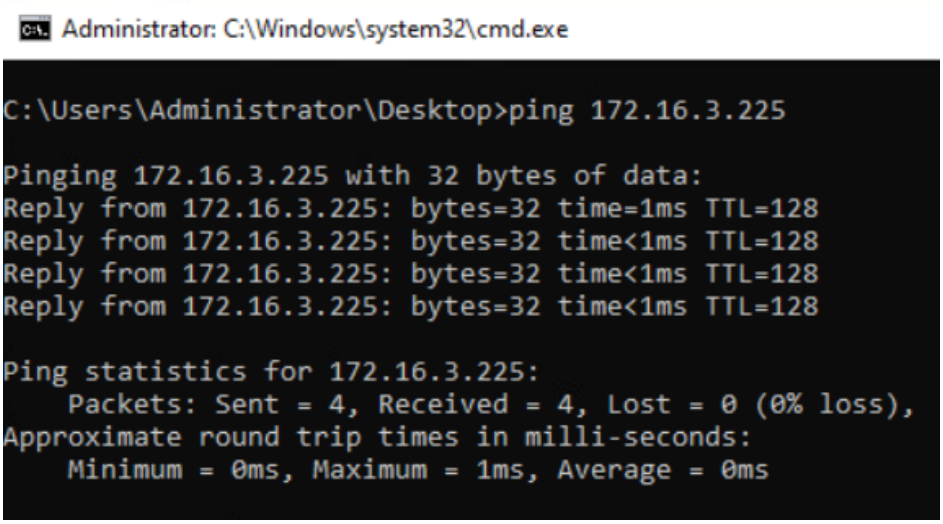

Figure 3: Pinging the "win-xp-18" machine from "win-hunt-18"

There are numerous ways of logging into the Windows XP machine, such as exploiting the BIOS, using Metasploit, leveraging FTP, or employing Netcat. For this lab, I will be focusing on FTP and Netcat to access the Windows XP machine from my Windows 10 system. I began by logging into a Linux machine, specifically a Kali Linux environment, and opened Zenmap to examine the Windows XP machine for any open ports or potential backdoors (ways to get in). Zenmap is particularly helpful for both beginners and advanced users because it simplifies the setup of various Nmap scans (such as the “Slow and Comprehensive” profile) and provides easy-to-read results, as shown in Figure 4. Moreover, Kali Linux comes pre-loaded with many penetration testing tools, making it a convenient platform for quickly scanning and enumerating open ports. After entering the target IP address of the Windows XP machine, I discovered several suspicious services: a Doom server at port 666, a VNC service at port 5900, an FTP

server, and a backdoor listening on port 6666, as shown in Figures 5-7. Finding an FTP server on a legacy OS like Windows XP often indicates a higher chance of misconfiguration or weak credentials that make it easy to access; FTP transmits data in plaintext by default, which also raises security concerns. Similarly, the presence of a backdoor on port 6666 means that I can attempt to connect using Netcat, a versatile networking tool that can read and write data across network connections. Netcat is frequently used for quickly exploring open ports, checking for vulnerabilities, transferring files, and even establishing shells if the target service allows it. These discoveries highlight various security risks on the Windows XP machine and underscore the importance of using tools like Zenmap to routinely assess and secure network hosts. It also means that I can leverage the FTP server to access the Windows XP machine and potentially exploit the open backdoor with Netcat, further demonstrating how an attacker might circumvent

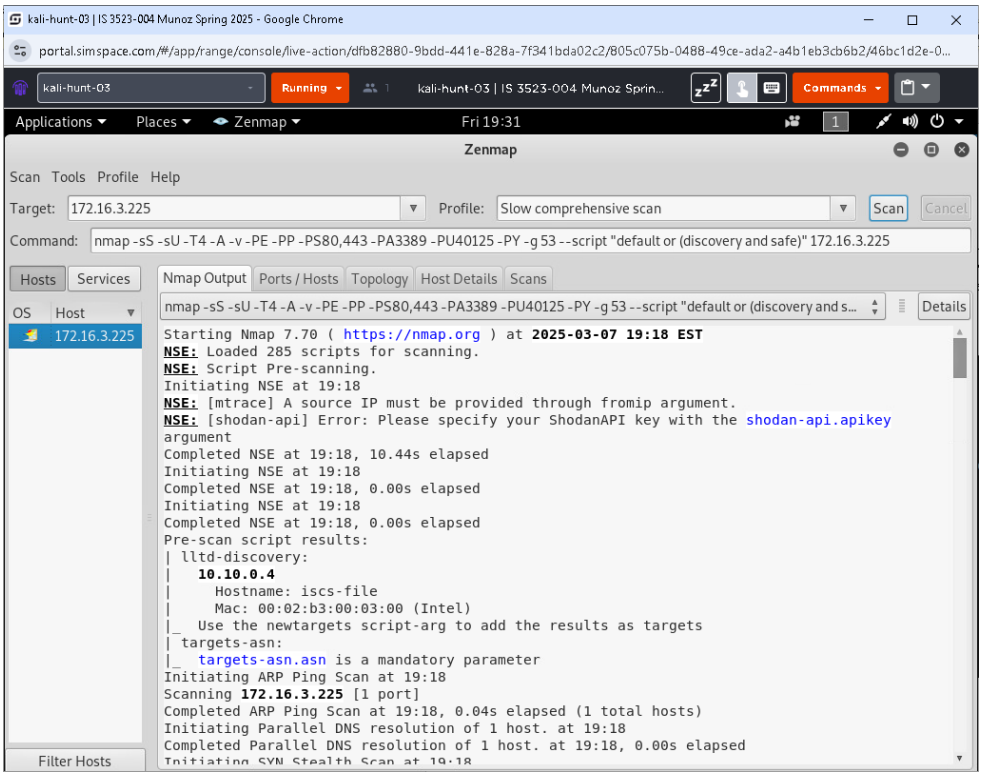

Figure 4: Zenmap of "172.16.3.225" using the "Slow comprehensive scan" profile

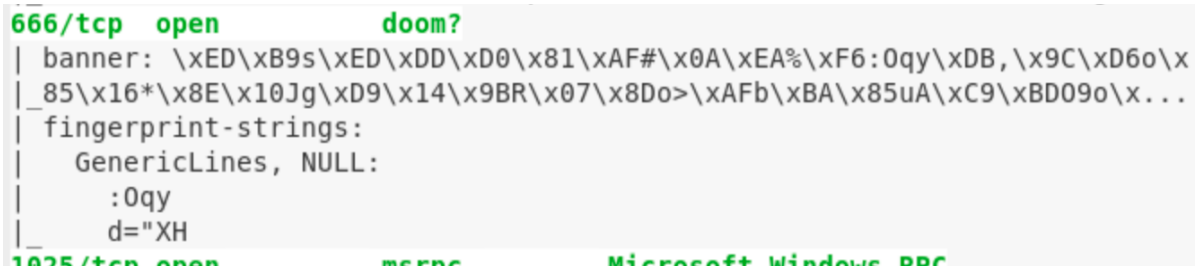

Figure 5: Zenmap of Doom on port 666

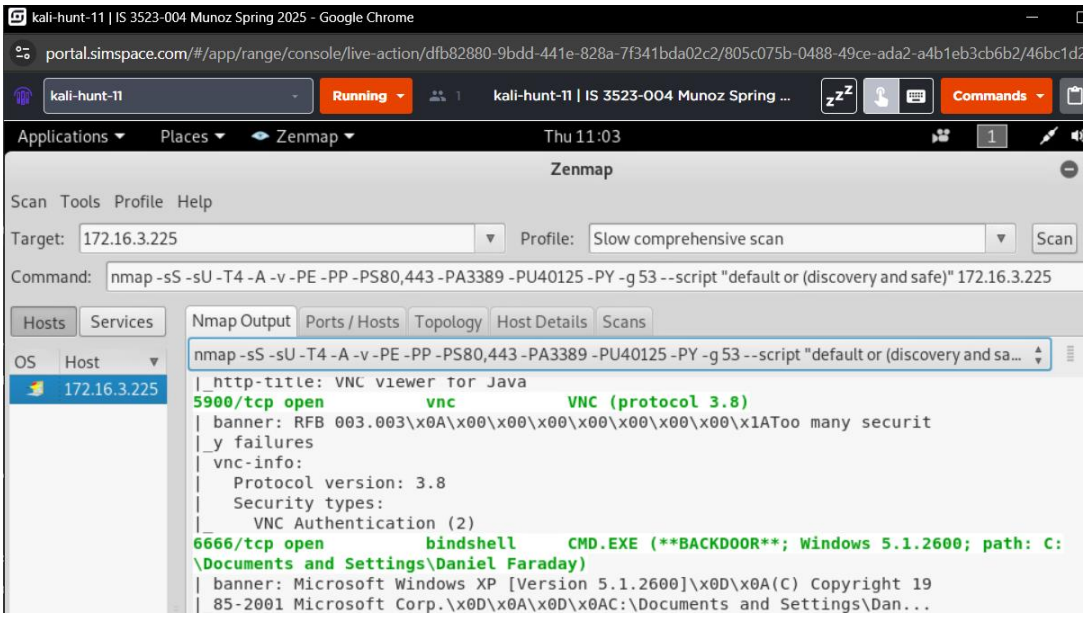

Figure 6: Zenmap figure of VNC on port 5900 and backdoor on port 6666

```
PORT     STATE         SERVICE       VERSION
21/tcp   open          ftp           Microsoft ftpd
|_banner: 220 Microsoft FTP Service
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
| 05-25-10  04:23PM                   56 lock.bat
| 03-07-25  11:09PM           1573434382 memory.690b700d.7z
| 12-29-04  12:07PM                61440 nc.exe
| 05-13-10  04:12PM                 3429 Razor.1911.IRC.nfo
| 05-25-10  06:03PM                77824 runasspc.exe
|_05-25-10  05:55PM       <DIR>          VNC4
| ftp-syst:
|   SYST: Windows_NT
|   STAT:
| Microsoft FTP Service status:
|      Connected to 172.16.3.102
|      Logged in as IEUser@
|      TYPE: ASCII, FORM: Nonprint; STRUcture: File; transfer MODE: STREAM
|      No data connection
| End of status.
```

Figure 7: Zenmap figure of the FTP server operating on the Windows XP machine

I then proceeded to access the FTP server from the Windows 10 machine. I opened up the terminal and I used command, “ftp 172.16.3.225”, to access the FTP server. I used the default username of “anonymous” and default password of “password” to login. I then proceeded to list the files within the FTP server folder which was also shown in Zenmap but I listed it again to make sure I was in correctly as shown in Figure 8. This means I was successfully able to access the Windows XP Machine through FTP. As you can see, there are multiple files which we will go over more in depth at a later point but spoiler alert, lock.bat was used to lock all the users out of their accounts so the adversary could play video games and sell video games and movies. This was a prime example of what the previous adversary should have done to Professor Kaufman. The previous adversary in lab 1 did not DDOS Professor Kaufman so he was able to notice what was going on and take actions however in this situation, Daniel Faraday could not because he was locked out of his system.

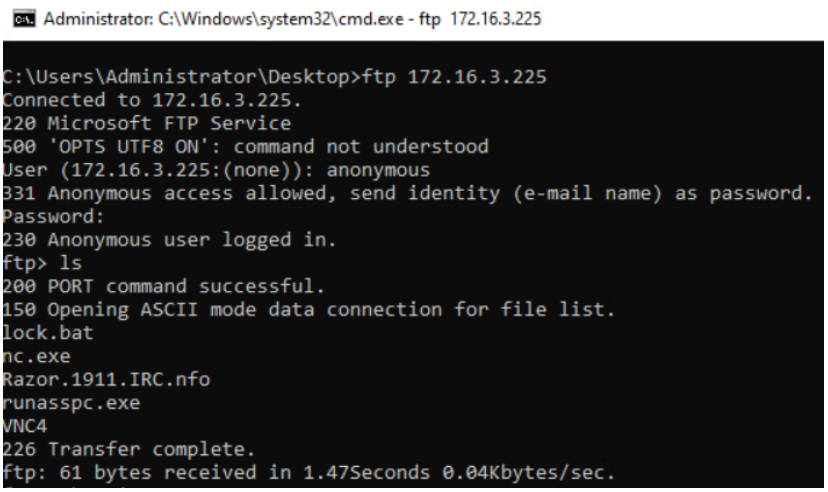

Figure 8: FTP into the Windows XP Machine from the Windows 10 machine terminal

I then proceeded to completely access the Windows XP machine through the backdoor on port 6666 that I discovered via Zenmap as shown in Figure 9. Using Netcat (nc), I opened a connection to the IP address on this port, which immediately presented me with a Windows XP command prompt, indicating that a backdoor shell was actively listening for incoming connections. Because the shell ran with privileges that allowed user management, I was able to execute command, “net user "Daniel Faraday" sqz137”, to reset the user’s password as shown in Figure 9. This action effectively granted me control over that account on the compromised system. Gaining access in this way underscores how dangerous an unauthorized backdoor can be—especially on an older operating system like Windows XP, which is no longer receiving official security updates. By listening on an unguarded port, this backdoor facilitated a direct channel to execute commands, manipulate user credentials, and explore or modify the filesystem. Tools like Netcat are powerful due to their ability to read and write raw data across the network, making them ideal for everything from simple port testing to establishing fully interactive shells when a service is misconfigured or intentionally left open. Through this unprotected port, I bypassed the standard authentication mechanisms entirely, demonstrating how critical it is to routinely scan and harden network services to prevent such intrusions.

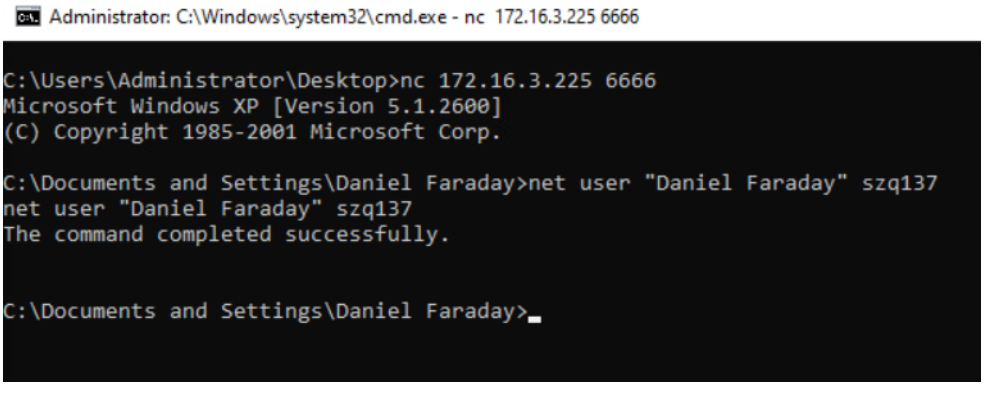

Figure 9: Using Netcat to access the Windows XP machine through the backdoor on port 6666

Imaging the System

Once I got in, I wanted to create a .img file of the system so that I could run it through various forensic and analysis tools, allowing me to better understand what was happening at a deeper level within Windows XP. Taking a forensic image is often the first step in a thorough investigation because it captures a snapshot of the system in its compromised state, making it easier to perform offline analysis without altering critical evidence. Professor Munoz had previously sent an email listing several Windows XP, compatible tools that would be especially useful for this lab; however, I must have been out of the room during office hours when he explained their usage. I reached out to a classmate for the information, and he promptly forwarded me the email as shown in Figure 10, ensuring I had everything I needed to proceed with the imaging process.

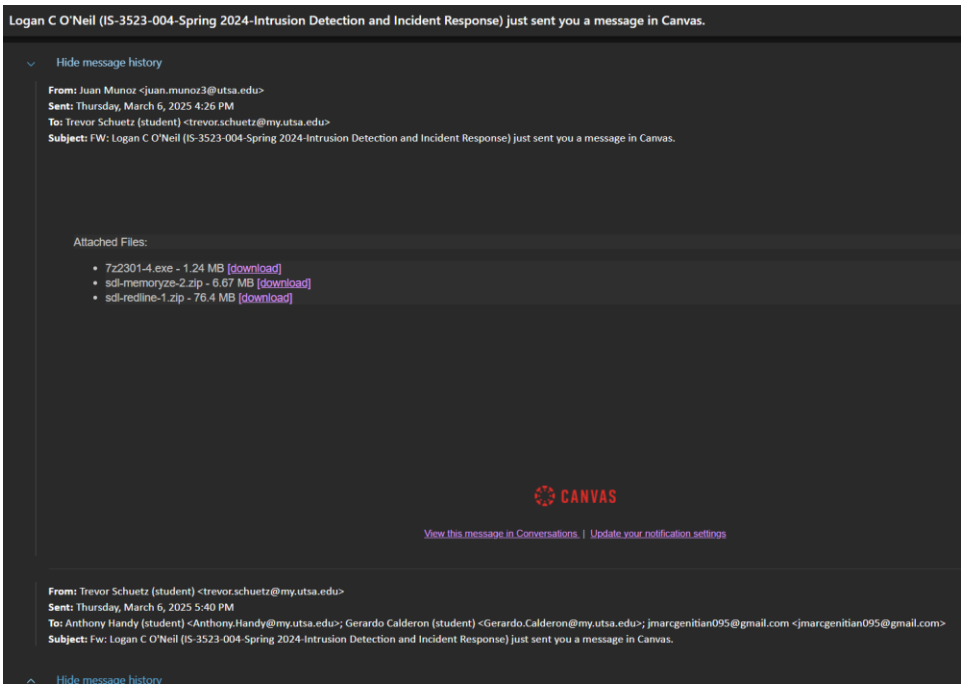

Figure 10: Downloads for 7zip, Memoryze, and Redline

I then uploaded all of the files I had downloaded into the File Management tab in SimSpace as shown in Figure 11, which serves as a convenient, centralized location for managing files across multiple virtual machines. Using the “create link” button in the File Management interface generates a unique URL, enabling me to open Internet Explorer on any target VM and instantly download the files I need. In this lab, I am installing Redline and Autopsy on the Windows 10 machine, given that these tools require more modern operating system libraries and significant resources to run effectively. Redline, often associated with Mandiant, specializes in memory forensics by capturing and analyzing a system’s running state; it can identify malicious processes, injected code, and other indicators of compromise within the live memory capture. Autopsy, built on The Sleuth Kit framework, is an open-source digital forensics platform that offers a broad range of analysis capabilities, from examining disk images and parsing the registry

to reconstructing user activity timelines. Conversely, the Windows XP system will host Memoryze and 7zip. Memoryze, another Mandiant tool, excels at creating memory images on older or compromised systems, producing the .img files that investigators subsequently analyze. Meanwhile, 7zip helps compress these large .img files to expedite their transfer from the Windows XP machine over to the Windows 10 environment, where Redline and Autopsy then perform a more detailed forensic examination. This ensures that we are able to successfully understand what was going on within the Windows XP system without just simply manually going through the file system. Although, it should be noted that it is important to manually go through the file system to get an understanding of what was going on and find possible artifacts.

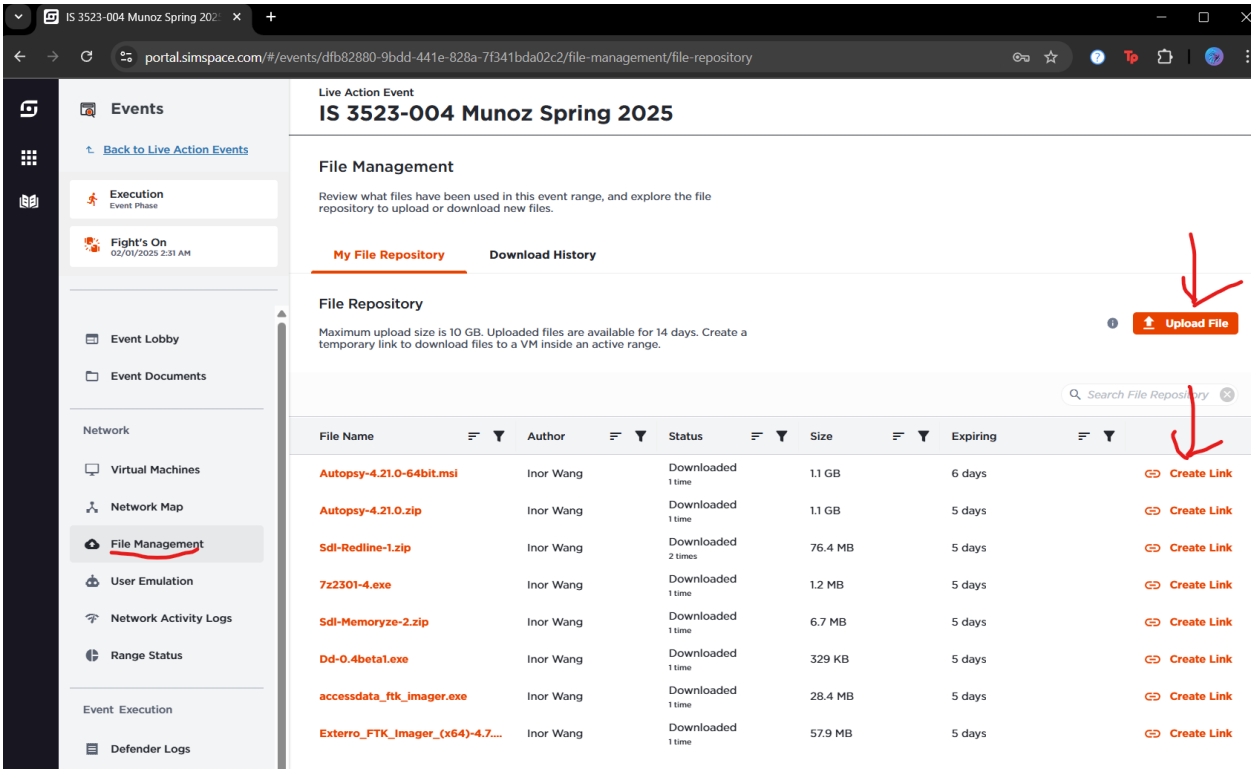

Figure 11: File Mangement tab within SimSpace

Once I finished downloading Memoryze, I organized all of the required files into a single directory on the Desktop as shown in Figure 12, ensuring that everything I needed was in one place for easy access. From there, I ran the "MemoryDD.bat" file, which is the core script responsible for capturing a snapshot (.img file) of the entire system's memory. This process, which took about ten minutes, resulted in a forensically sound .img file being saved in the Audits folder as shown in Figure 13. The file is approximately 3.1 GB which is the correct size as noted by Professor Munoz during lecture. Obtaining a dedicated memory capture is a critical step in digital forensics, especially in the context of investigating a compromised Windows XP system, because volatile data such as running processes, open network connections, encryption keys, and injected malicious code often resides only in memory. If this information is not preserved promptly, it can easily be lost whenever the system is powered down or rebooted. Memoryze is particularly useful for this task, as it is designed to work on older operating systems while

generating a detailed memory image that can then be analyzed offline with tools like Redline or Autopsy. By securing an .img file before taking any further investigative steps, you essentially freeze the system's current state for deeper scrutiny, helping ensure that key evidence is not altered or erased.

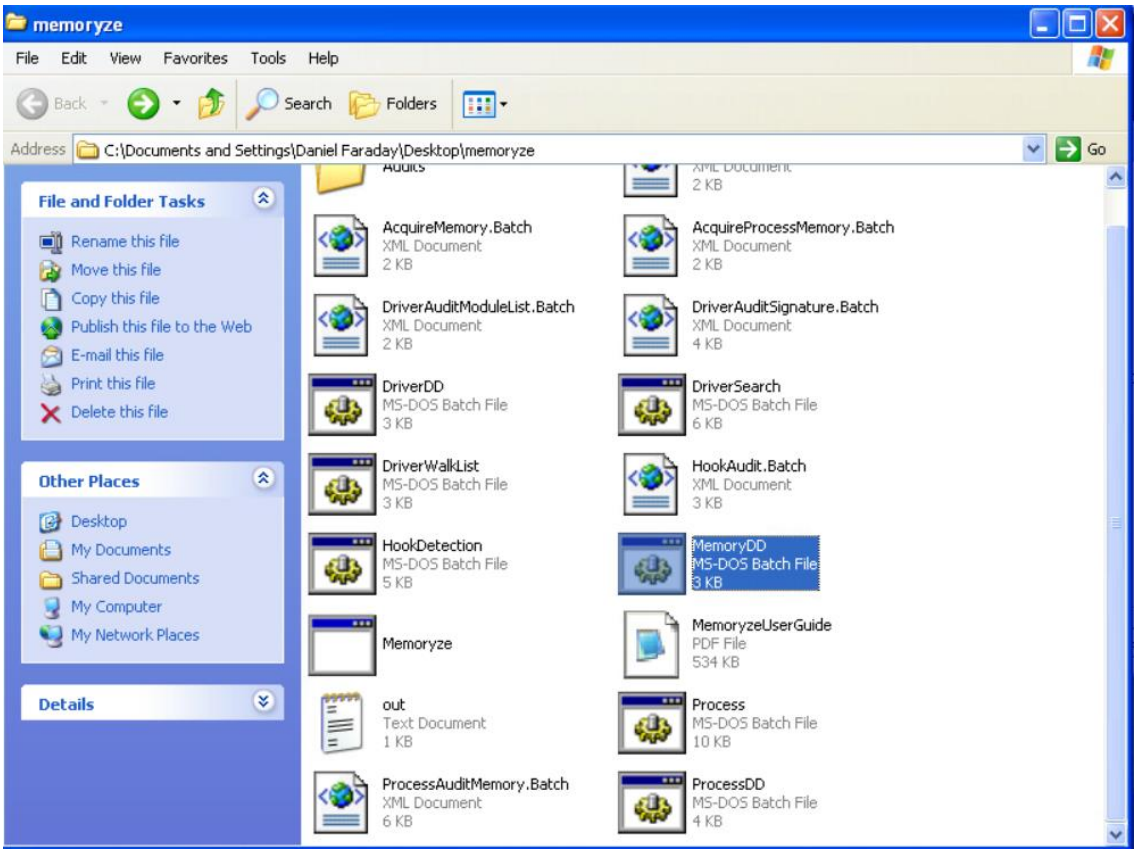

Figure 12: Memoryze folder, specifically "MemoryDD.bat"

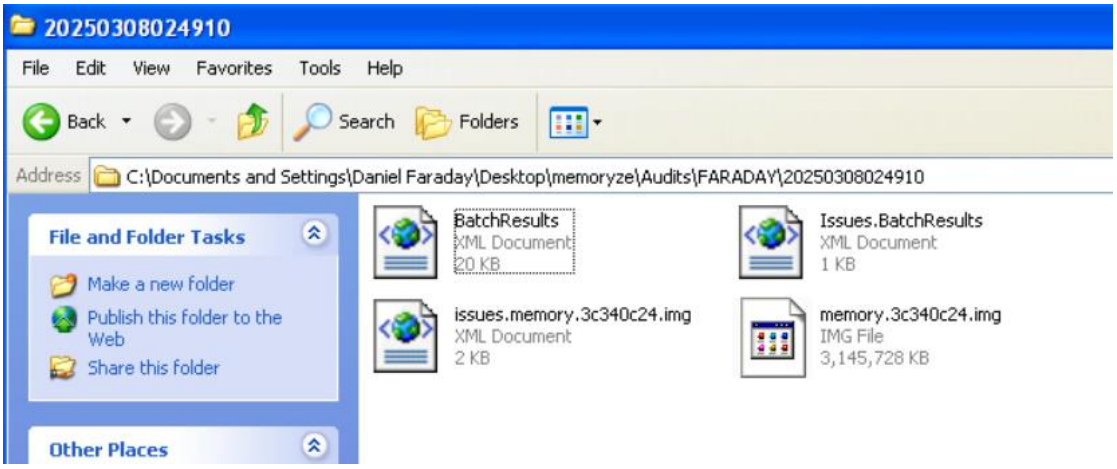

Figure 13: Successful completion of "MemoryDD.bat" which exported an .img file

I then proceeded to compress the folder containing the .img file using 7zip as shown in Figure 14, a crucial step to make transferring the image to the Windows 10 machine faster and more efficient. Data captured in a raw .img format can be extremely large, especially for memory-intensive processes, so using a high-compression tool like 7zip greatly reduces the file size without compromising the integrity of the forensic data. By minimizing the file size, not only does the transfer consume less bandwidth, but it also shortens the time needed to move files between virtual environments, an important consideration when working with potentially unstable or limited lab resources. This compression step ultimately ensures that I can quickly proceed to the more robust analysis phase on Windows 10, where tools like Redline and Autopsy are better equipped to process and interpret the memory capture.

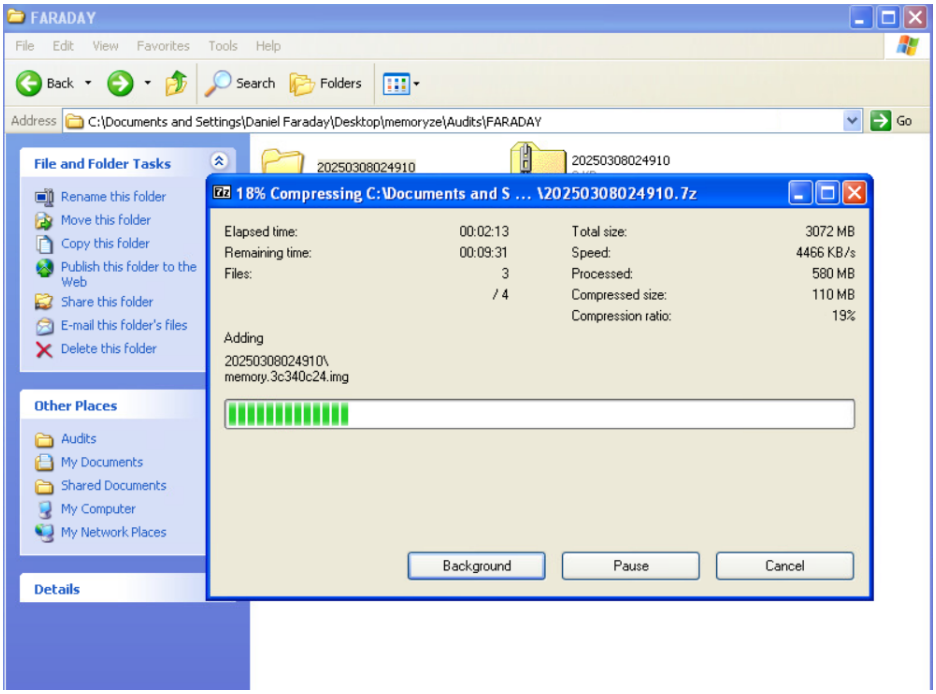

Figure 14: 7zipping the .img file

After creating and compressing the .img file on the Windows XP machine, I placed that newly created .7z archive into the FTP server directory, specifically located at C:\Inetpub\ftproot, to facilitate an easy and direct transfer to the Windows 10 machine as shown in Figure 15. This directory is where the FTP service on Windows XP automatically looks to serve or retrieve files, and by dropping the file there, I could connect from the Windows 10 system using the built-in ftp client. As shown in the screenshot, I logged into the FTP server at 172.16.3.225 using an anonymous account from the Windows 10 machine, which confirms that anonymous logins had not been disabled. Once connected, I used standard FTP commands to list the files in the root directory, confirming that my .7z archive was indeed present alongside other files such as lock.bat and nc.exe. I then issued the get command to retrieve the archive from the XP machine as shown in Figure 16. Despite the simplicity of FTP (and its inherent security limitations like unencrypted credentials), it proved effective for quickly transferring large files within this

controlled lab environment. As soon as the file transfer was complete, I could then unzip and analyze the memory image on the more capable Windows 10 system.

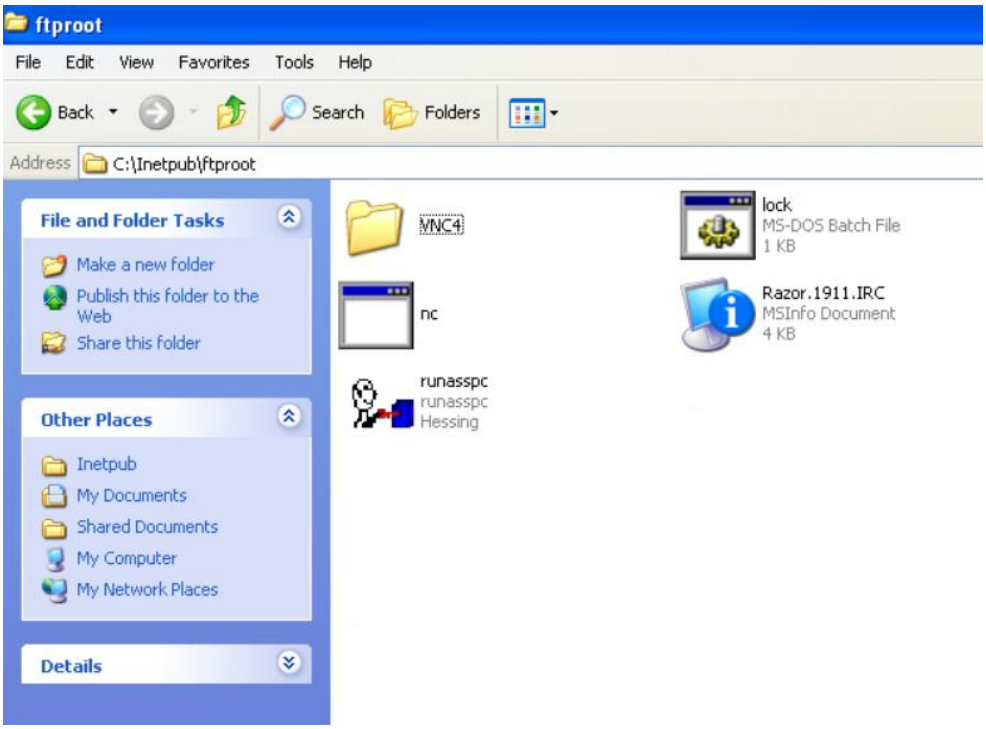

Figure 15: "ftproot" folder

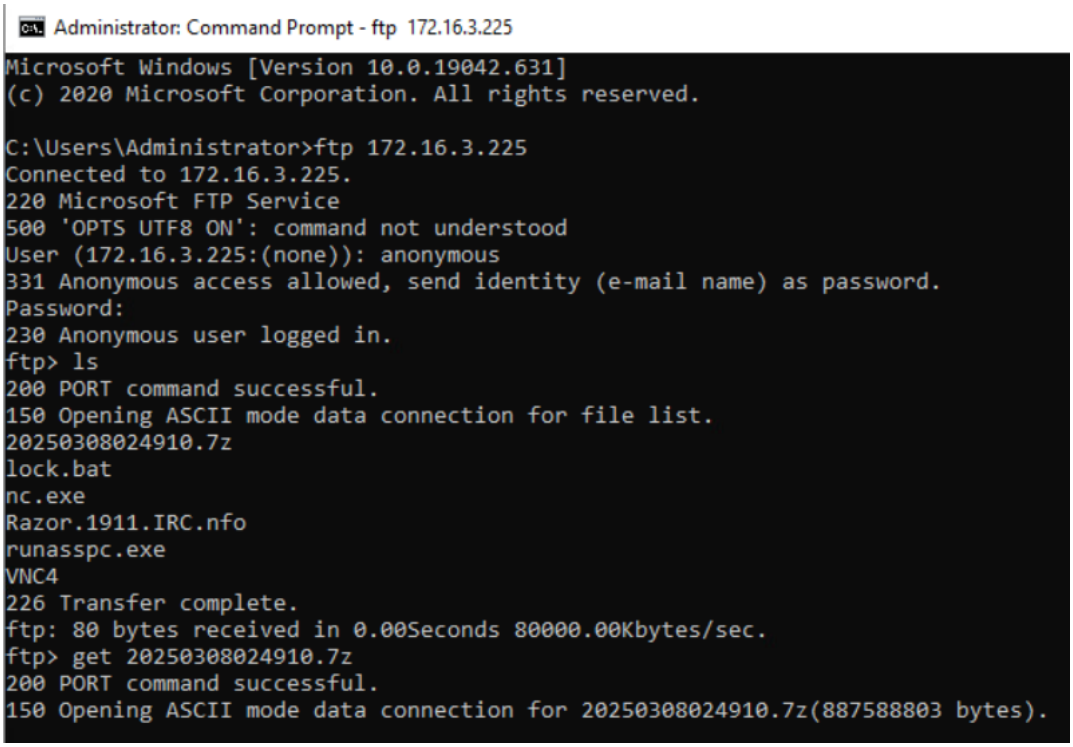

Figure 16: Access the FTP folder from Windows 10 to extract the zipped .img file

Since I connected to the Windows XP machine via FTP from the C:\Users\Administrator directory in Windows 10 as shown in Figure 16, the downloaded file would be placed into that directory. Once the transfer was complete, I unzipped the folder and placed the extracted folder into my “InorLab2” directory on the Desktop as shown in Figure 17. With the raw .img file now present in the Windows 10 environment, I was able to proceed with the next phase of memory analysis and forensics using tools that are more compatible and resource-capable than those available on Windows XP.

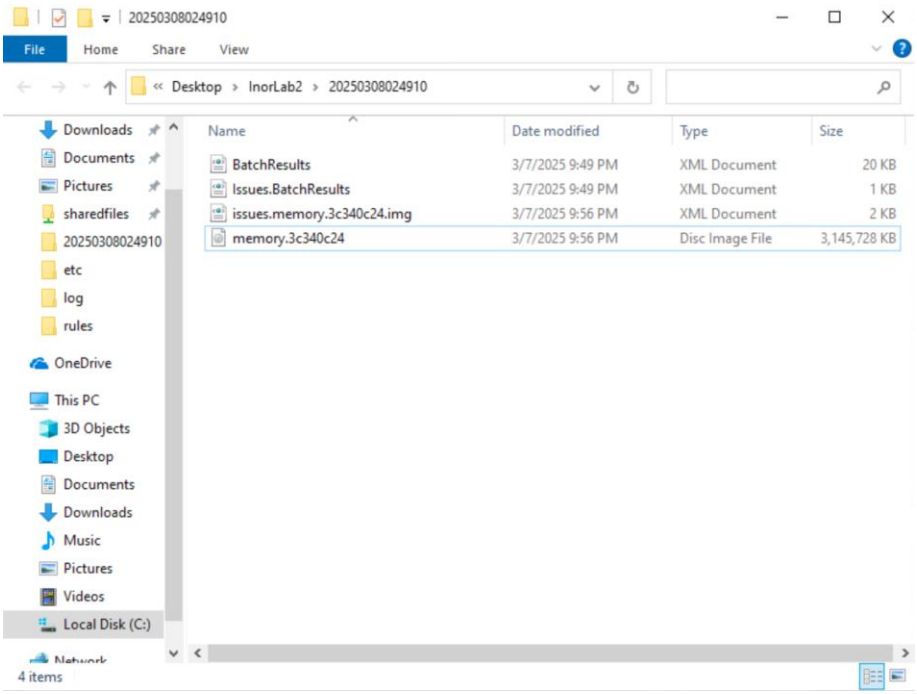

Figure 17: Extracted .img file within Windows 10

Methodology

I then downloaded Redline and Autopsy into the Windows 10 Machine. The process was noted in the "Accessing the Compromised VM" section of the lab. I then proceeded into both tools and imported the .img file into both to start analyzing as shown in Figure 18 and 19. The analyzation process for the tools takes a long time. For reference, the Redline program took approximately twenty-five minutes, and the Autopsy program took approximately one-to-two hours.

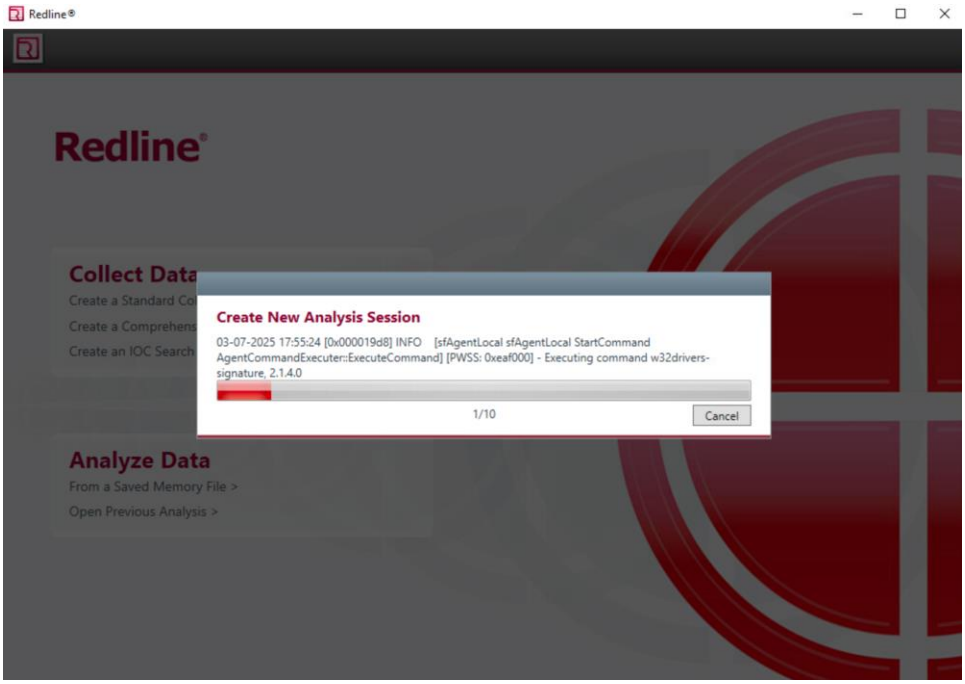

Figure 18: Redline analyzing the .img file

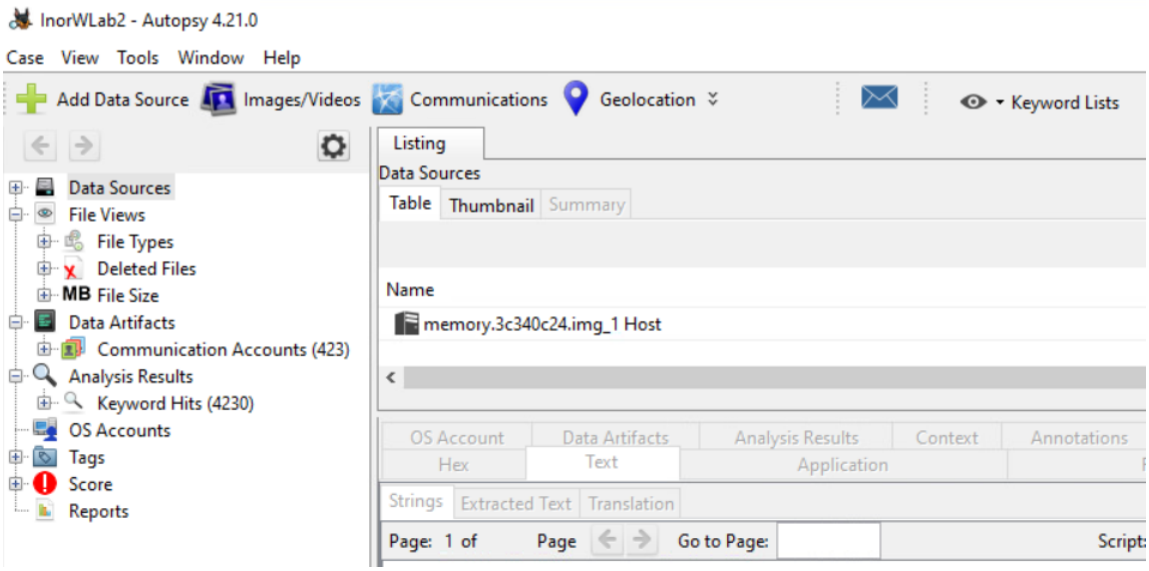

Figure 19: Autopsy analyzing the .img file

As the programs were actively analyzing the .img files in the background, I decided to make the most of my time, just as a professional incident response analyst would in a real-world scenario. Rather than waiting idly for the tools to finish processing, I logged back into the Windows XP system to conduct a manual examination of the file system. This hands-on approach allowed me to search for potential artifacts, suspicious files, hidden directories, unusual programs, or documents that might not be immediately flagged by automated analysis tools. By exploring the system manually, I hoped to gather additional context and insight into the user's behavior, any unauthorized changes made to the system, and the overall timeline of compromise. Manual investigation complements memory and forensic analysis by revealing signs that might otherwise go unnoticed, such as altered file paths, recently modified documents, or remnants of malware activity left behind. This proactive step not only helped reinforce my understanding of what went wrong, but also mirrored the real-world mindset of staying productive and thorough during every phase of an investigation.

Firstly, I went to the C:\ drive to start manually looking through the file system as shown in Figure 20. I went into the Documents and Settings directory first. I did not find anything out of the ordinary within the directory however I found the administrative tools. I went through each tool to hopefully understand a little bit more. The only one I personally found useful was Event Viewer, specifically the Application logs. I noticed that there were a lot of errors coming from the source of "crypt32" as shown in Figure 21. I noted that the date was during our lab so it may have been one of the students however it is important to note. These repeated crypt32 errors, all occurring within a short time span, may suggest persistent issues related to the Windows cryptographic services, possibly due to failed certificate validation, corrupted system files, or unauthorized tampering. I also noticed a suspicious source with the name of "DrWatson", so I clicked on it to learn more. The description of the event showed that quote, "The application, C:\hxdefrootkit\hxdef100.exe, generated an application error". This event log entry from Dr. Watson reveals that a program named hxdef100.exe, located in the directory C:\hxdefrootkit\, encountered an application error. The exception code generated was c0000005, which typically indicates an access violation, meaning the program attempted to read or write to protected memory, often a sign of instability, malicious behavior, or poorly written code. The name and directory of the application strongly suggest that this program is Hacker Defender (hxdef), a well-known rootkit used to hide files, processes, and registry entries from the user and system tools. Rootkits like Hacker Defender are often used by attackers to maintain persistent, covert access to a compromised system while avoiding detection. The fact that it was active on the system and triggered a crash logged by Dr. Watson is a significant finding, confirming that stealth malware was installed and likely being used to conceal other activities.

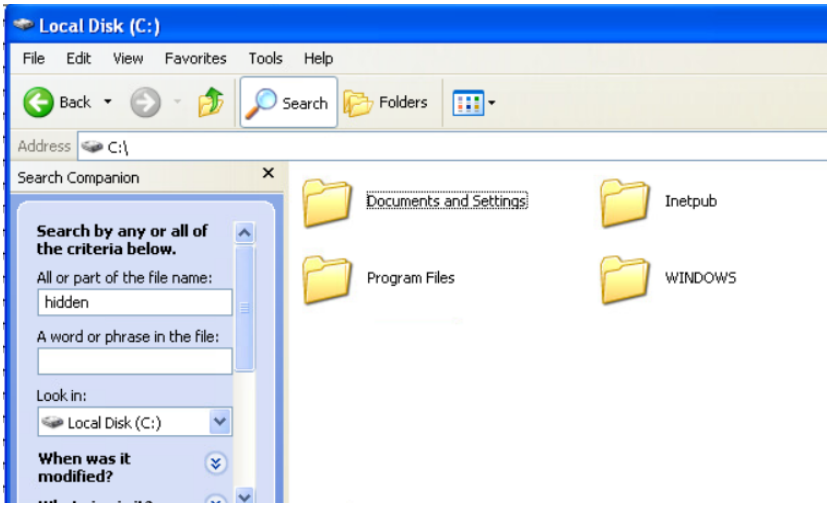

Figure 20: C:\ drive

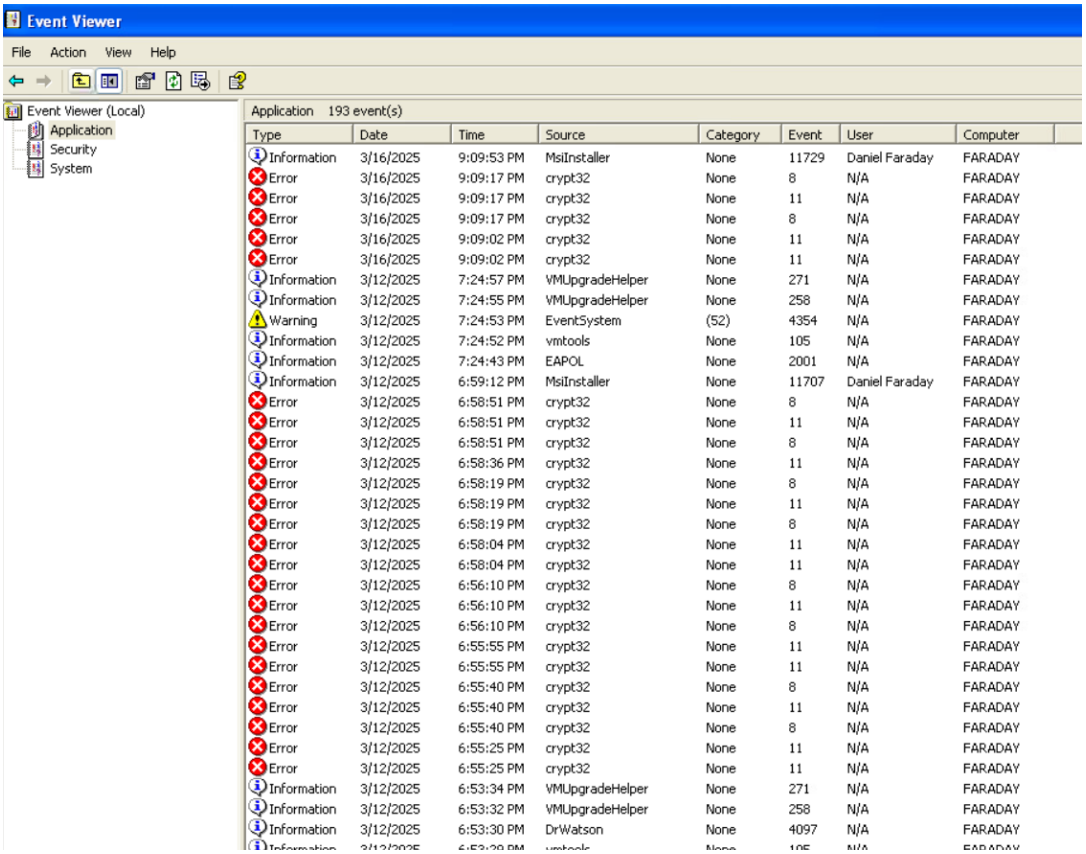

Figure 21: Event Viewer of the Windows XP machine

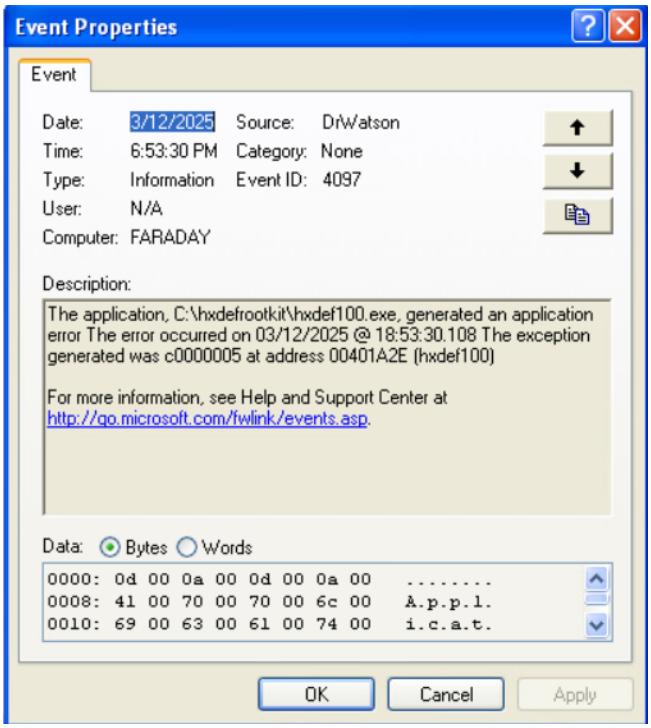

Figure 22: Suspicious event within the Event Viewer

Within the C:\Program Files\ directory, I noticed that there were a lot of video games in the MSN Gaming zone directory as shown in Figure 23, which is unusual because this system should not have video games.

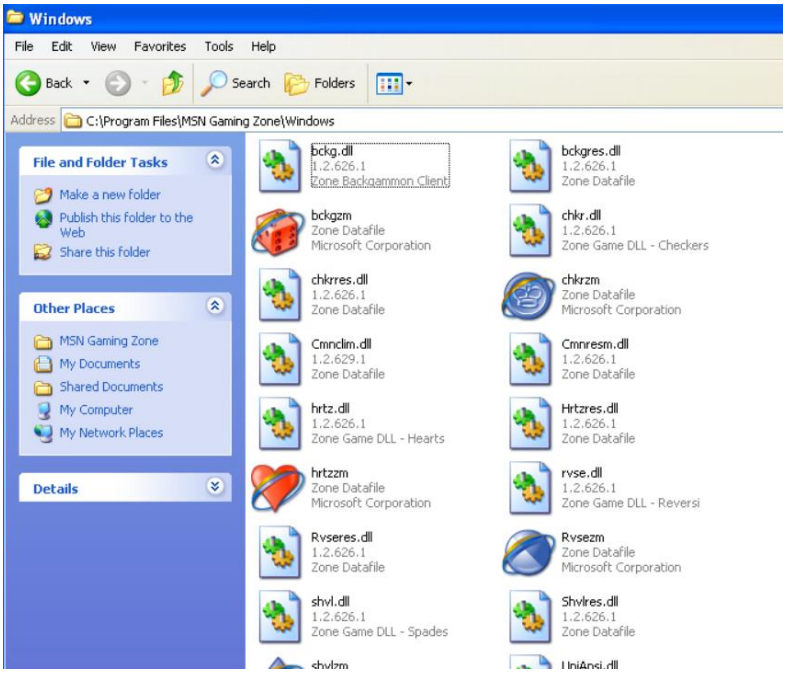

Figure 23: MSN Gaming Zone directory

I then went to the Logfiles directory within the system32 directory. I noticed that there were two folders, “MSFTPSVC1” and “W3SVC1”. I went into the MSFTPSVC1 folder and there were three log text files. I opened the first log file and it showed that someone logged into the FTP server with the username of Anonymous, a password of fake@fake.com, and created two files: “files.tar” and “7za.exe” as shown in Figure 24. files.tar is most likely a container for malicious files similar to a zipped file and 7za.exe is used to extract those files, bypassing user interaction. The file “files.tar” is particularly suspicious, as .tar is a common Unix-based archive format, short for "Tape Archive." While it is not inherently malicious, .tar files are frequently used to bundle multiple files together for easier distribution or storage, making them a convenient way for attackers to hide payloads or package malware components into a single folder. These folders can contain executable files, scripts, or other malicious artifacts that would only be revealed after extraction. Combined, they represent a highly suspicious behavior, suggesting an attacker was trying to deploy tools or extract hidden contents on the compromised system.

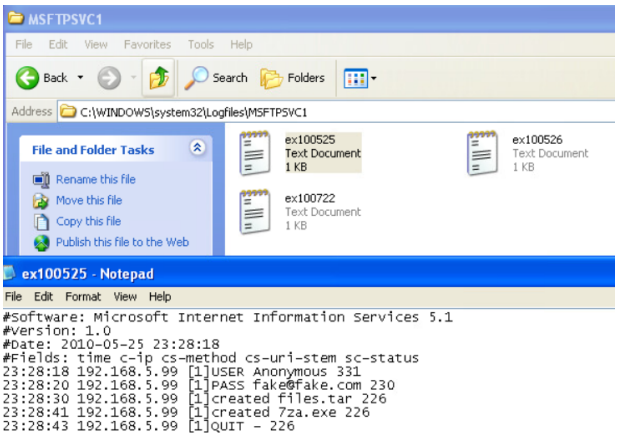

Figure 24: "ex100525" log file for FTP

I then proceeded to the ftp folder and noticed that there were multiple files as shown in Figure 25, all of which may have been within the files.tar. As you can see, there is a nc.exe file within the FTP folder. nc.exe is the Windows executable version of Netcat, a powerful and versatile networking tool often used by both system administrators and attackers alike. In a legitimate setting, Netcat can be used for troubleshooting and testing network services, but in the hands of a malicious actor, it becomes a tool for covert operations. One of its most dangerous capabilities is the ability to create a listener on a specific port, essentially opening a backdoor that provides remote access to the compromised system. This aligns with what I observed earlier in the investigation when I used Zenmap to scan the network and detected a suspicious open port: 6666. Upon further exploration, I was able to use Netcat to connect to this port, confirming that a backdoor shell was listening and could be interacted with through remote commands. Given this, it is highly likely that the attacker initially accessed the system through the misconfigured FTP server, which allowed anonymous logins, and uploaded several files, including nc.exe. After gaining a foothold, the attacker probably executed nc.exe on the victim machine to establish persistent, remote access via port 6666. This method not only gave the attacker full control of the

machine but also bypassed standard login mechanisms, making detection and prevention more difficult. From there, the hacker may have changed the password of Daniel's user account. The password change is only a speculation, but we can further investigate if the password was truly changed or not.

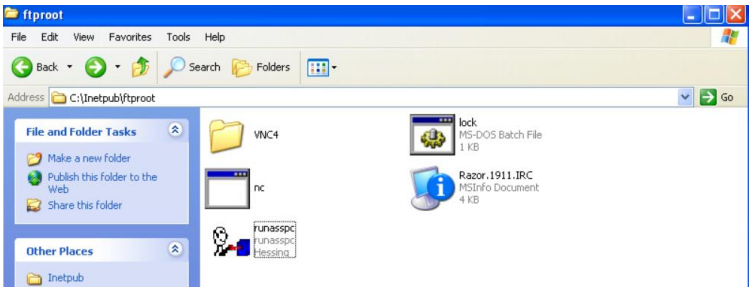

Figure 25: FTP folder

I discovered a file named lock.bat. Curious about its purpose, I right-clicked on the file and selected the Edit option to inspect its contents. Inside the script, I saw three simple lines of code: @echo off, rundll32.exe user32.dll, LockWorkStation, and cls as shown in Figure 26. This immediately indicated that the batch file was designed to lock the current user session. Specifically, the second line is the key command, it uses rundll32.exe to call the LockWorkStation function from user32.dll, which forces the Windows XP system to lock the screen and return to the login prompt, just like pressing Windows Key + L. The inclusion of @echo off and cls is purely cosmetic, used to suppress command-line output and clear the terminal after execution. While the script itself is not inherently malicious, in the context of this compromised system, it clearly served a purpose. It was likely used by the attacker to kick out the legitimate user (Daniel Faraday), making it harder for anyone to interrupt or detect further malicious actions taking place, such as file transfers, privilege escalation, or backdoor creation. This simple script effectively functioned as a denial-of-access tool, buying time for the attacker to operate undisturbed.

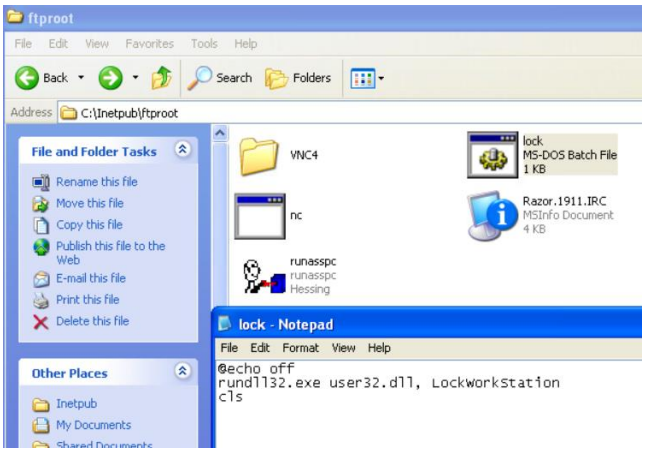

Figure 26: lock.bat script

Then I noticed a file named Razor.1911.IRC in the ftproot directory, which immediately caught my attention due to its association with Razor 1911, a well-known software cracking and piracy group. Curious about its contents, I opened the file and discovered that it was a text-based README containing installation instructions for a pirated update of The Sims 3 (v1.2.7) as shown in Figure 27. The file included promotional game notes, compatibility warnings for other cracked versions like those from WARG, and steps for applying the update, such as extracting .RAR files, installing the patch, copying the crack into the game directory, and running the game. While the file itself was not executable or malicious, it strongly suggested that the system had been used to download, install, or distribute pirated software. This is especially concerning because pirated games and their cracks are often bundled with malware like keyloggers, trojans, or remote access tools. The Razor file, when viewed alongside other suspicious tools on the system such as nc.exe, runasspc.exe, and lock.bat, paints a picture of a machine that was either compromised or being actively used for unauthorized activities, including software piracy and potential malware deployment.

Figure 27: Opened the Razor.1911.IRC file in Notepad

I went back to the log files and went to the second log file as shown in Figure 28. I noticed that the user logged in through the same username and password, and created the “runasspc.exe” within the FTP folder. I went back to the FTP folder and noticed the executable. It stood out among the other files because of its unique icon and unfamiliar name. Curious about its purpose, I decided to run it and observed that it launched a program called RunAsSpc, a third-party utility used to run applications under different user credentials, typically allowing a standard user to execute programs with administrator privileges. The application immediately displayed a console output and a GUI pop-up indicating that it was "FOR PRIVATE USE" and referenced a commercial license from a German website, suggesting it was not intended for enterprise or unauthorized use as shown in Figure 29. Interestingly, it attempted to open a file named crypt.spc but returned an error message saying “File could not be opened.” This .spc file is likely an encrypted credential container or configuration file that RunAsSpc relies on to securely store and reuse admin credentials without exposing them in plaintext. The failure to open this file indicates the tool was either misconfigured or incomplete, but its presence is a major red flag. In the context of this compromised environment, the attacker likely uploaded runasspc.exe in an attempt to escalate privileges or run additional tools silently, bypassing the need to manually log in as an administrator. Combined with other tools in the same folder, like nc.exe, lock.bat, and Razor.1911.IRC, this suggests a deliberate effort to establish control over the system, maintain persistence, and carry out further malicious activities undetected.

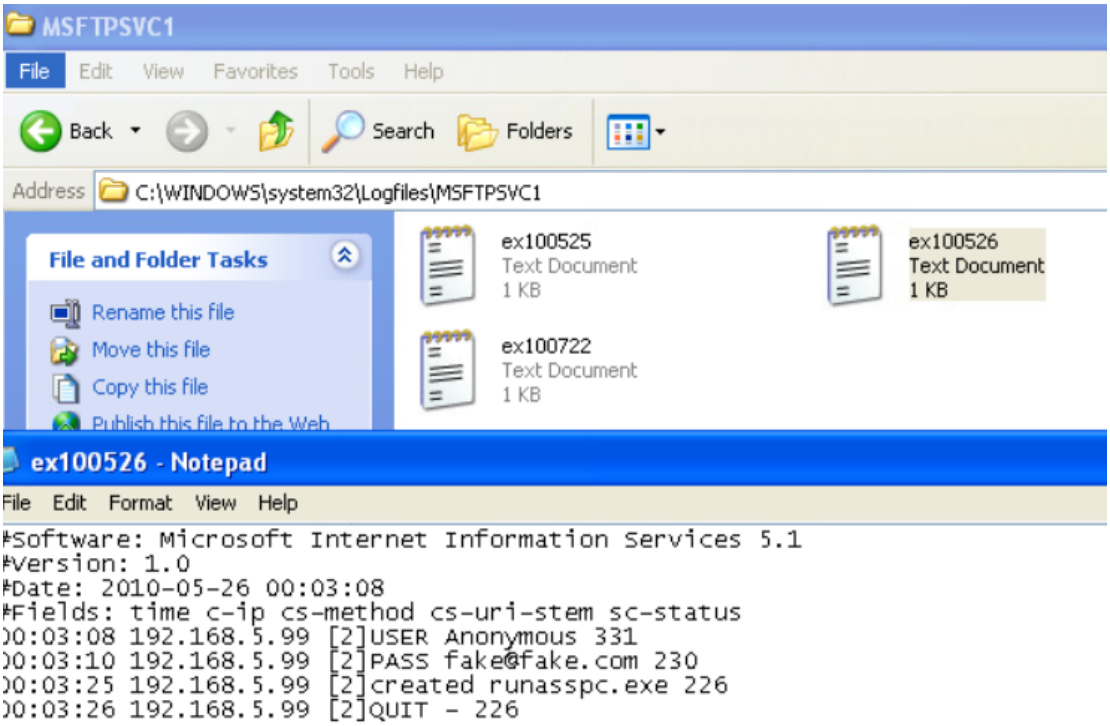

Figure 28: Second log file

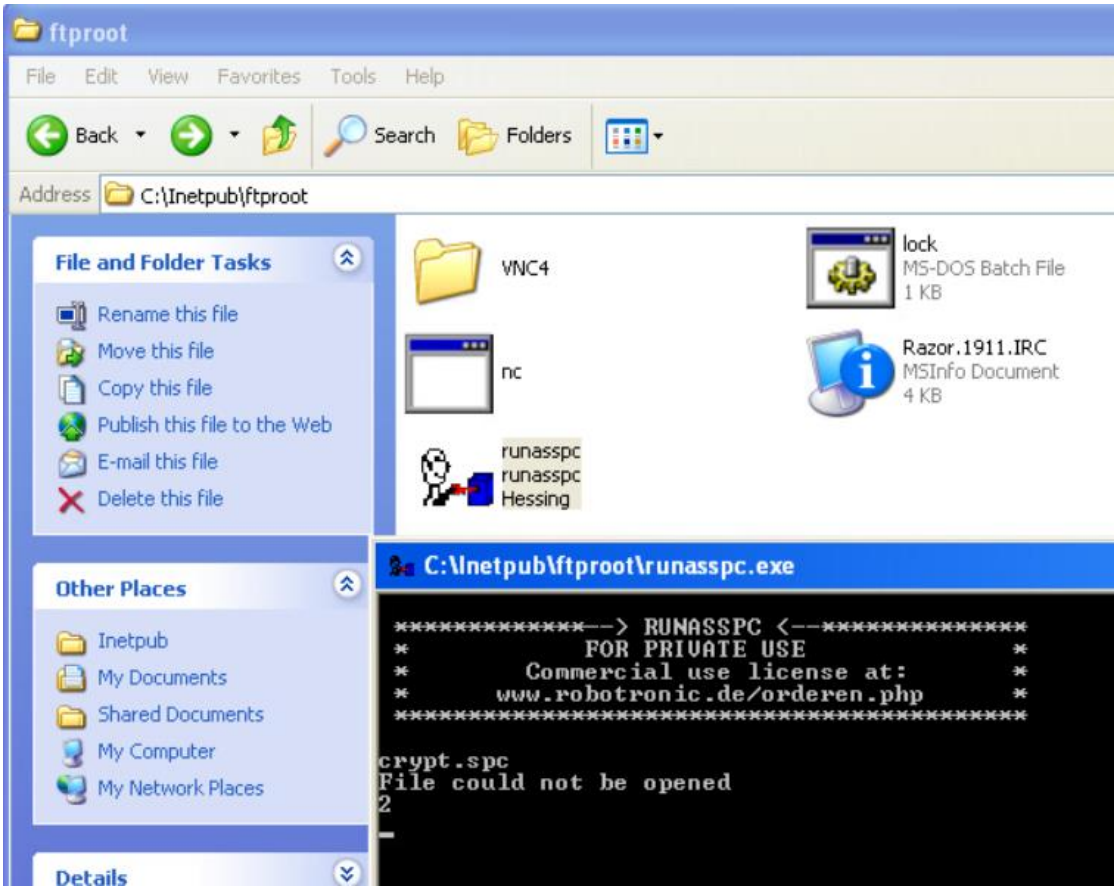

Figure 29: runasspc.exe within the FTP folder

I then noticed the VNC4 folder within the FTP folder. VNC is a remote desktop protocol. It allows a user to view and control the desktop of another computer. This is a big red alarm and may be another hint to how the hacker was able to access this computer. I used the search function within Windows XP and I searched for "vnc". Then I discovered a particularly concerning file named startup.bat located in C:\Documents and Settings\Daniel Faraday\Startup, which is a directory that automatically runs any executable or batch file upon system login as shown in Figure 30. I don't know how I missed this script as I was looking through the C directory however thankfully, I was able to find it. I opened the file and found that it contained commands to launch three separate programs: nc.exe, winvnc4.exe, and lock.bat. This confirmed that the attacker had established persistence on the system by scripting these tools to automatically execute every time the machine was rebooted. As I detailed earlier in the report, nc.exe is the Windows version of Netcat, which was used to open a backdoor listener on port 6666, providing command-line access to the system. The line nc.exe -L -p 6666 -e cmd.exe reinitializes that listener, allowing an attacker to connect remotely and gain shell access without needing to reconfigure anything manually. The second command launches winvnc4.exe, the VNC remote desktop software I previously analyzed in the VNC4 folder, granting the attacker full visual control over the desktop. Finally, lock.bat, which I had also examined earlier, uses a rundll32 call to lock the workstation, likely preventing local users from detecting or interrupting

malicious activities. The presence of this startup script ties together the attacker's method of initial access via FTP, their deployment of tools like Netcat and VNC, and the steps they took to maintain persistent, remote access while locking out legitimate users. It represents a classic example of a scripted intrusion with stealth, control, and automation all baked into one.

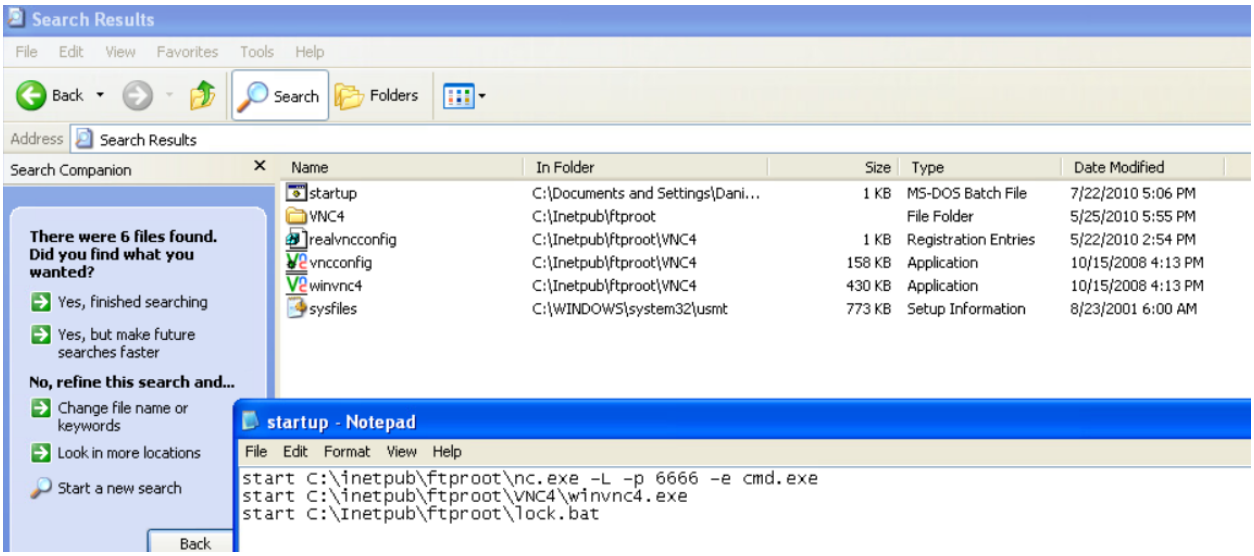

Figure 30: Startup script file

I then looked at the third and last FTP log and noticed that there were two logins with a different password of IEUser@ as shown in Figure 31. I also noticed that the log was two months later from the previous two logs showing us that some other user logged into the FTP server. I am not too sure what that user wanted to do in the system however it logged in through the backdoor that the previous user (hacker), established.

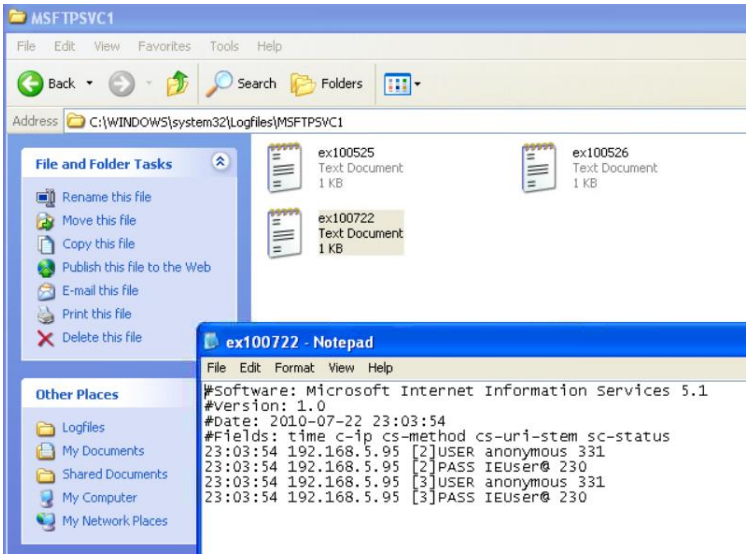

Figure 31: ex100722 log file

I then proceeded back to the Logfiles and proceeded to the W3SVC1 directory. This folder is use by Microsoft Internet Information Services (IIS) to store web server activity logs as shown in Figure 32. The folder name W3SVC1 specifically refers to the World Wide Web Publishing Service instance #1, which is responsible for logging HTTP requests made to the web server hosted on this machine. In the notepad preview, the log entry indicates that at 00:14:55 on May 26, 2010, the IP address 192.168.5.99 made an HTTP GET request to access /iisstart.asp, which is a default startup page for IIS. The status code 200 means the request was successful and the file was served correctly. This entry suggests that someone accessed the local web server hosted on the compromised Windows XP machine, likely just to test if the web server was up and running. While /iisstart.asp is harmless by itself, attackers often check this page to verify that the IIS server is operational before attempting to exploit vulnerabilities or upload malicious content.

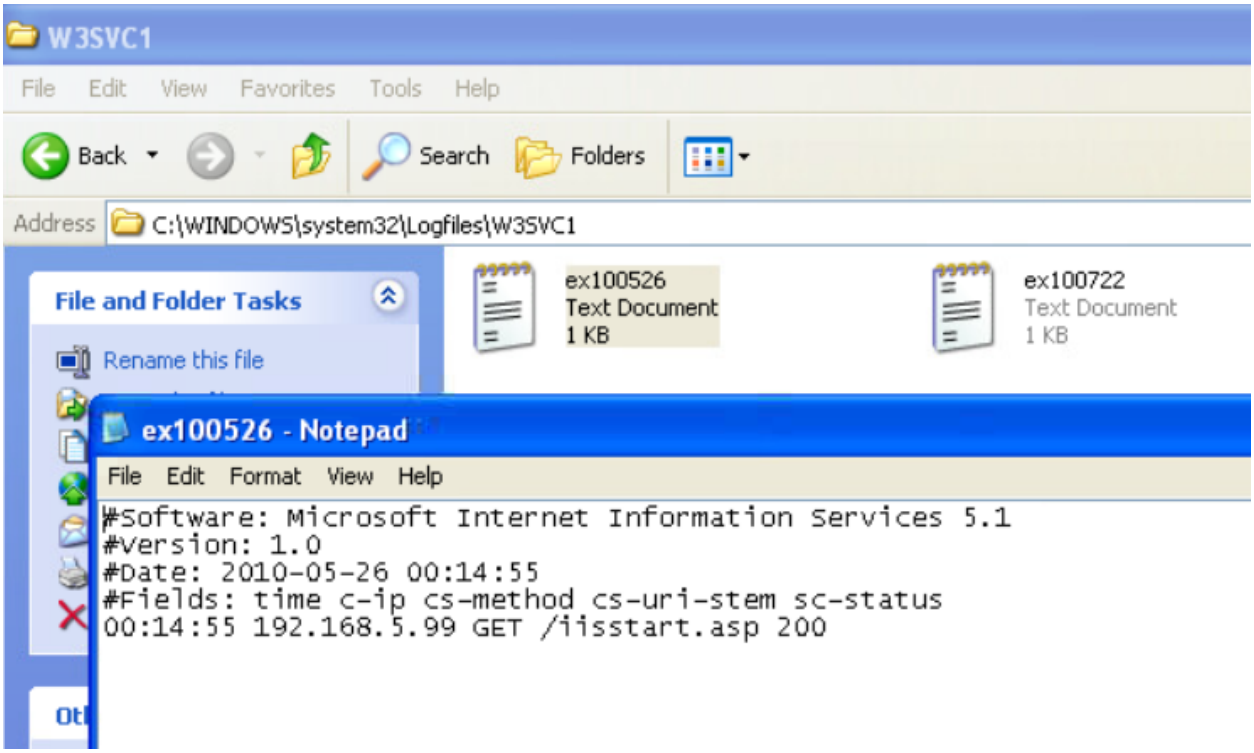

Figure 32: ex100526 log file within W3SVC1

I then proceeded to the second log file and it confirmed my suspicions. They did the same thing as last time although they also imported robots.txt and favicon.ico as shown in Figure 33. These logs indicate that someone (likely the attacker) was probing the web server hosted on the compromised Windows XP system to see what files or endpoints were available. The repeated access to iisstart.asp confirms that the IIS service was running properly, while the 404 errors for robots.txt and favicon.ico suggest this was a barebones or default IIS setup. In the context of everything else you've discovered (e.g., malicious scripts in the FTP directory, Netcat usage, VNC server), this log reinforces that the attacker was actively exploring multiple services (FTP, VNC, and HTTP) to maintain control or potentially escalate further.

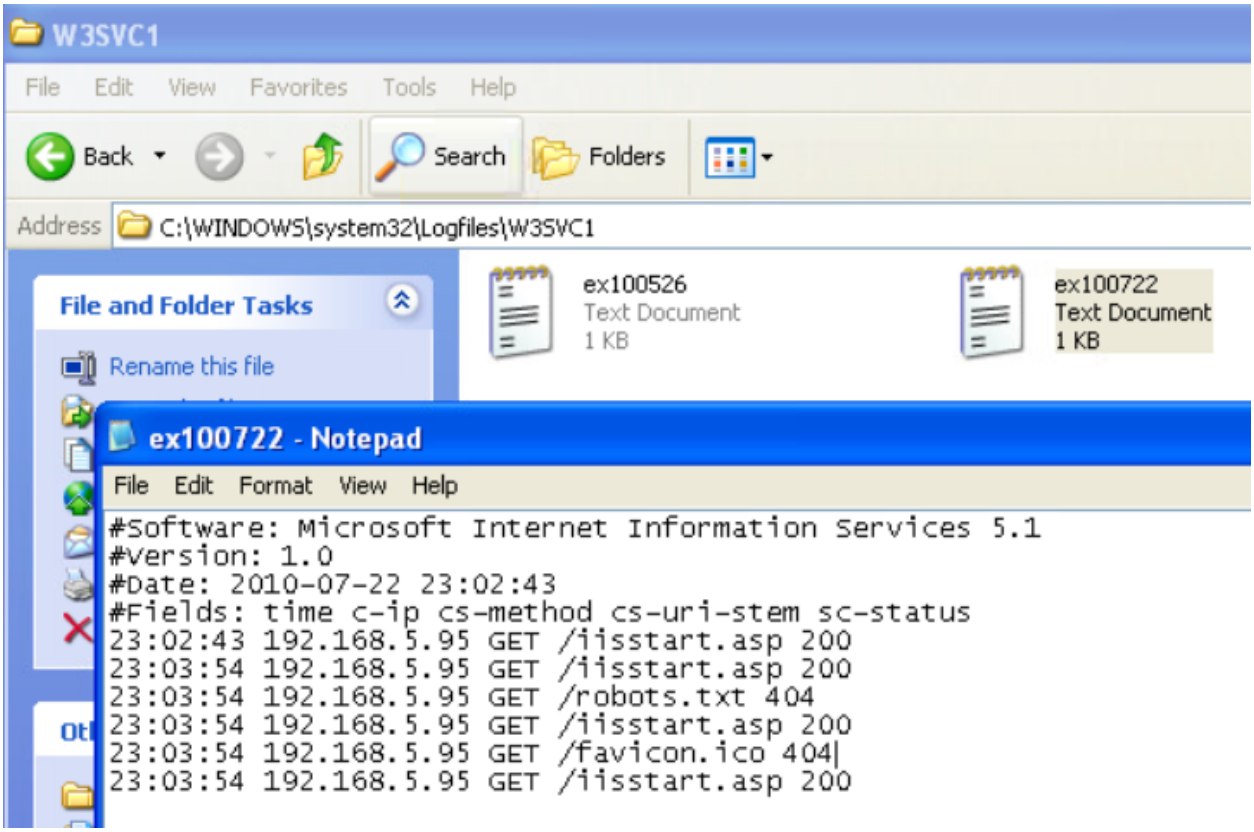

Figure 33: ex100722 log file within W3SVC1

I also noted that there were multiple video games within the Windows XP as shown before and as shown in Figures 34-35.

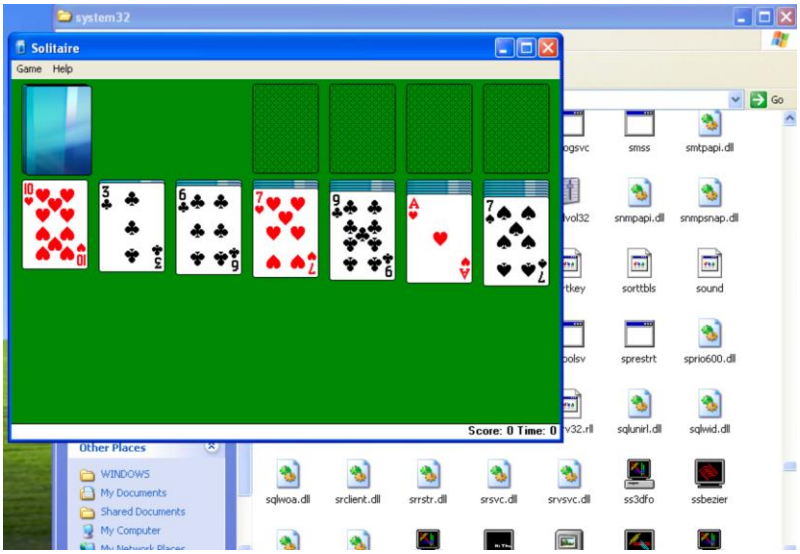

Figure 34: Game

Figure 35: Game

I then proceeded to RegEdit. I choose Software, Microsoft, WinNT, Current Version, and finally Winlogon. I noticed that the default password was Bond007 as shown in Figure 36, which is not Daniel's password and was changed from the hacker. I knew that the hacker changed the password, as noted before. I also searched it in Autopsy to verify as shown in Figure 37.

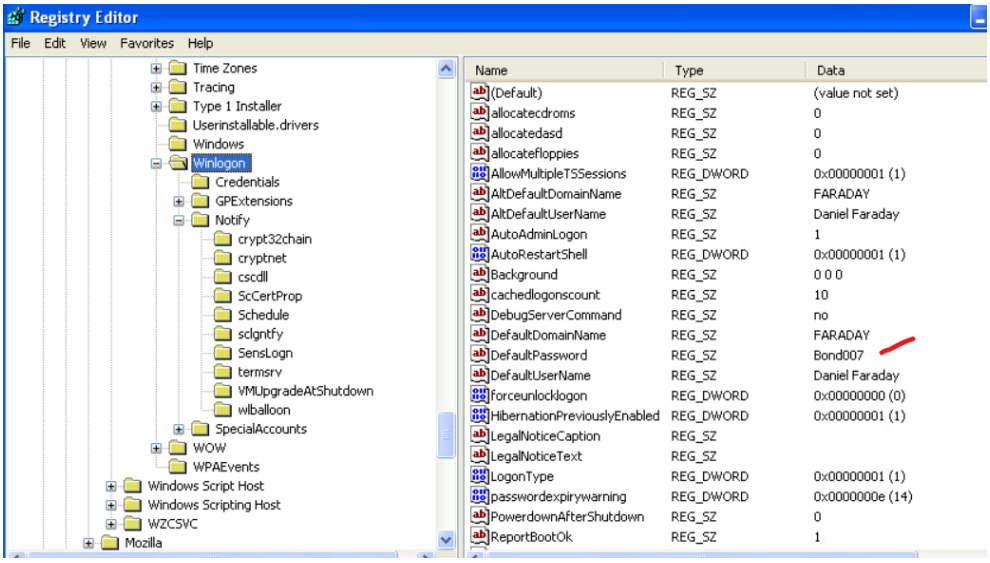

Figure 36: Regedit of the password change

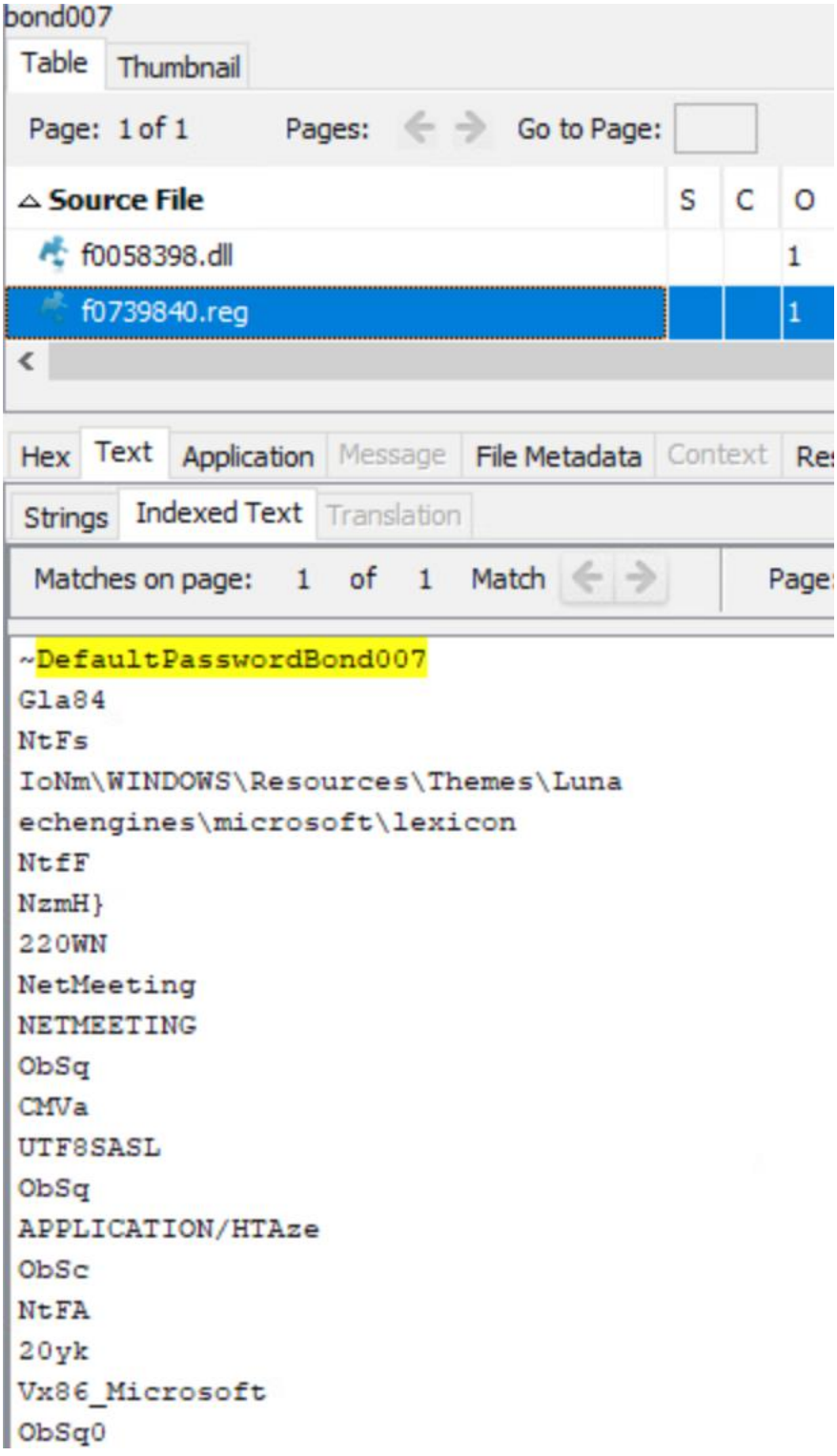

Figure 37: Autopsy of bond007

I then proceeded back to the Windows 10 machine, and I went to the Redline program. I went under processes, and I was looking through the processes. I noticed multiple things that I have noted before such as, vnc4, nc.exe, winlogon.exe, rundll32.exe, etc as shown in Figure 37.

However, I also noticed that there was a suspicious directory within the C: directory called “hidden” which wasn’t shown whenever I was manually looking through the Windows XP file system. This is most likely malicious so if I see any process occurring from that directory, I will immediately flag it. Some supcious things I found were, iroffer.exe, cryptcat.exe, hxdef100.exe, bircd.exe, and poisionivy.exe. After looking through these, I may be able to paint a better picture of what is going on.

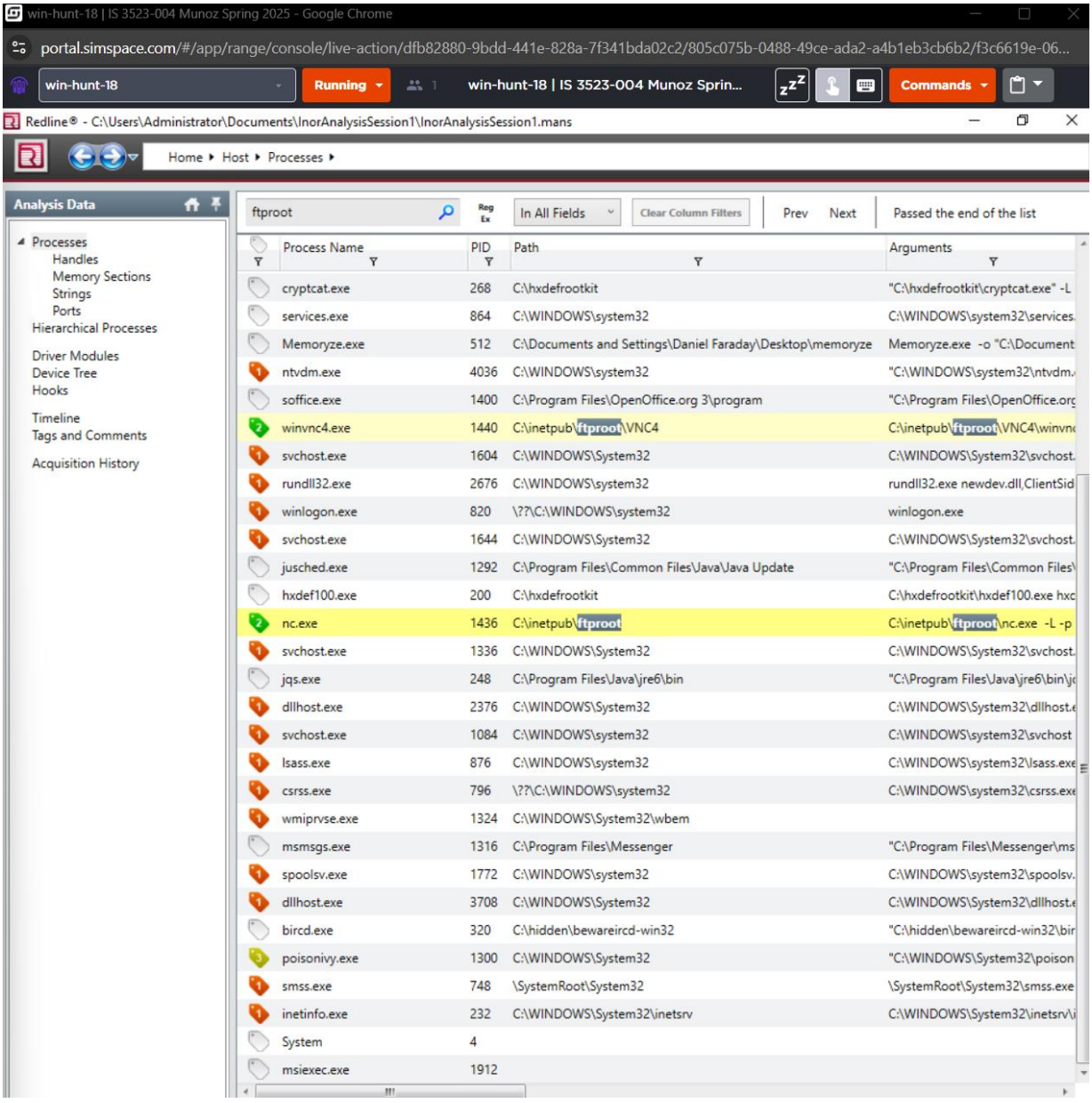

Figure 38: Redline output of the .img file of the Windows XP machine

While searching through the system, I came across a file located in the Prefetch directory named POISONIVY.EXE-0ED7FCCD.pf as shown in Figure 38. This immediately caught my attention because .pf files are Prefetch files created by Windows to optimize the loading time of applications. When a program is executed for the first time, Windows generates a

corresponding .pf file in the C:\WINDOWS\Prefetch\ directory, which logs various data about the file's execution and related file dependencies. The presence of a Prefetch file specifically for poisonivy.exe is extremely significant because it confirms that the executable was not only present on the system but was actually run. Poison Ivy is a well-known Remote Access Trojan (RAT) often used by attackers to gain full control over a compromised machine, including capabilities like keylogging, file transfers, remote shell, and more. Additionally, the associated .exe file was located in the C:\WINDOWS\System32\ directory, a common tactic used by adversaries to disguise malware as legitimate system components. The timestamp on the .pf file, dated July 22, 2010 at 1:28 PM, suggests the exact time Poison Ivy was executed, providing a valuable timeline marker. This discovery strongly supports the narrative that an attacker not only uploaded this malicious file but also actively ran it to establish long-term remote control over the system.

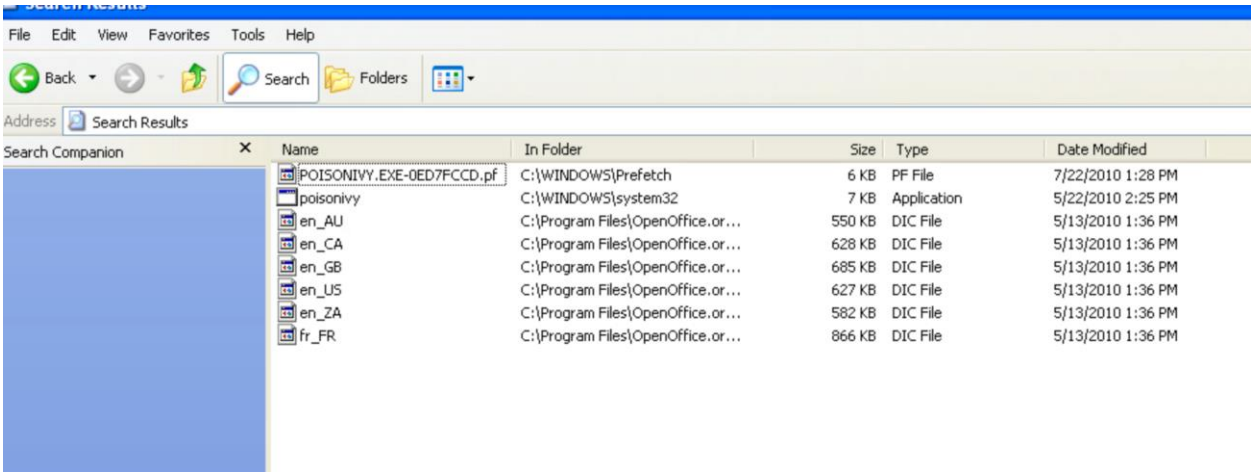

Figure 39: Poison Ivy search

Just like the Poison Ivy discovery, the presence of this .pf file indicates that the cryptcat.exe program was executed on the system. Cryptcat is a security tool similar to Netcat, but with the added capability of encrypting the data being transferred. It allows users, or in this case, attackers, to create encrypted tunnels for remote communication, making it harder to detect malicious data transfers using traditional network monitoring tools. The fact that Cryptcat was run strongly suggests that the attacker was trying to maintain stealth and confidentiality while communicating with or exfiltrating data from the system. This could have been used to securely transmit stolen files, establish an encrypted backdoor, or even interact with remote command-and-control servers without raising red flags. The use of Cryptcat also aligns with the broader pattern observed in the system, use of Netcat, VNC, FTP manipulation, and other utilities, demonstrating that the adversary was not only active but also intentional in covering their tracks and maintaining persistence.

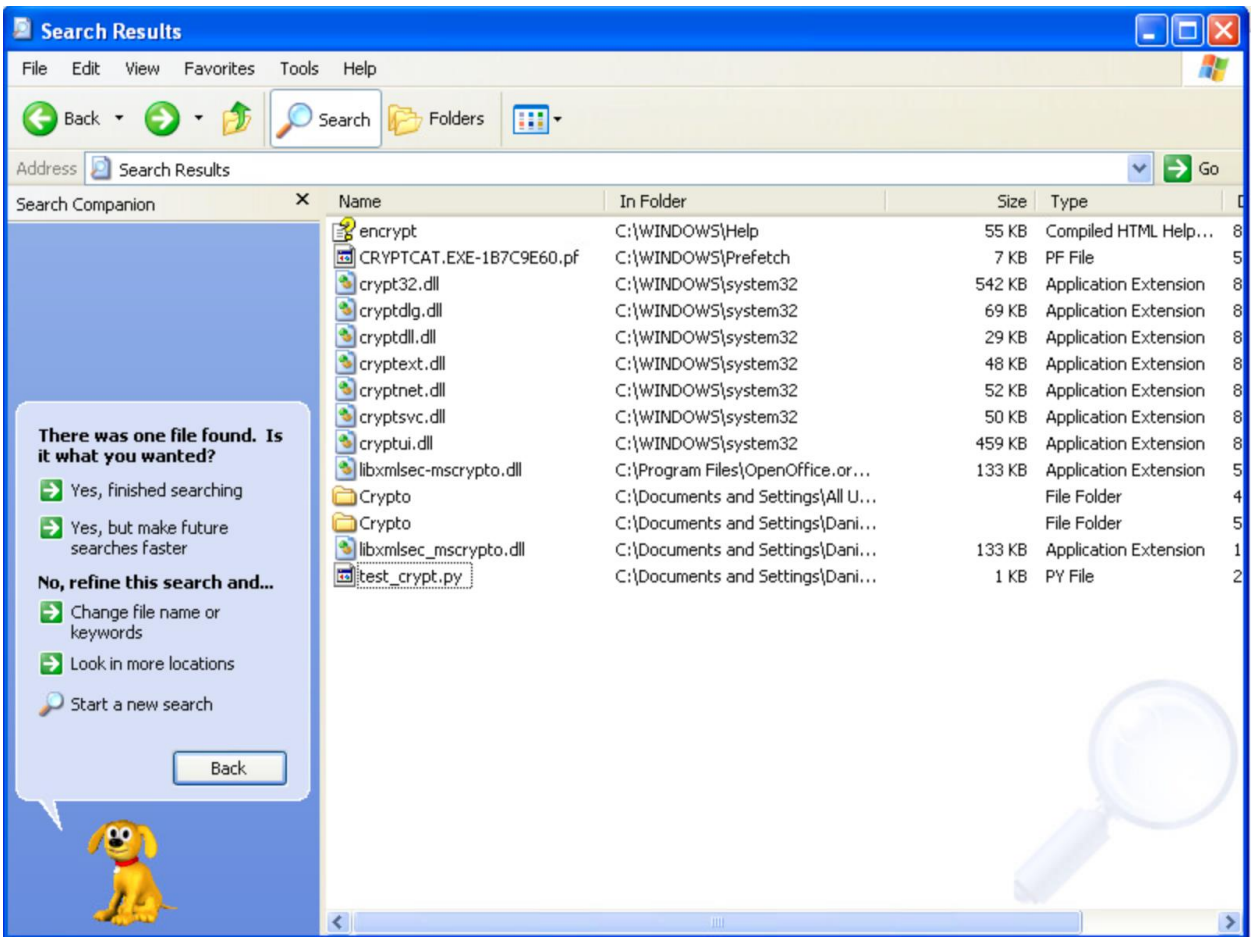

Figure 40: Cryptcat search

I also noticed that there was a Crypto folder with multiple RSA Machine Keys as shown in Figure 40. I initially thought these were crypto, the currency. However, I believe these may be the keys for the encrypted files that the hacker encrypted via Cryptcat.

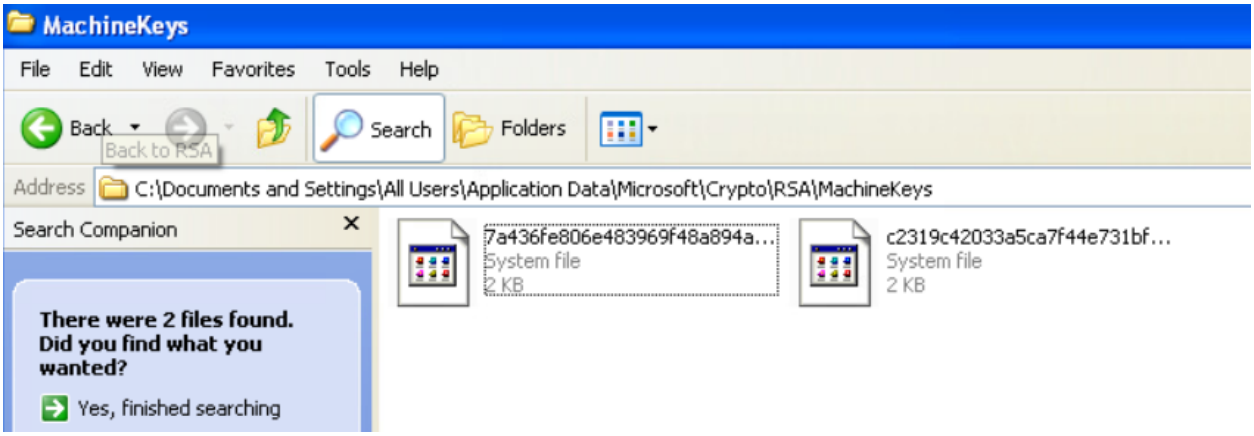

Figure 41: Crypto Machine Keys

Hxdef100.exe is a tool associated with the Hacker Defender rootkit, one of the more well-known user-mode rootkits for Windows. It can be used to hide files, processes, registry keys, and network activity, making malicious activity much harder to detect on an infected system. In this case, its presence suggests the attacker may have used it to conceal backdoors, malicious scripts, or tools like Netcat and VNC from security scans or system administrators. Bircd.exe, on the other hand, is a lightweight IRC server executable. Its use here likely indicates the attacker was running their own IRC server directly on the victim's machine, potentially to distribute files. Both tools point to an effort to maintain stealth and control over the system while supporting file sharing and remote access operations. It is further reinforced that it is a IRC server, I searched IRC within Autopsy and found this text snippet as shown in Figure 41.

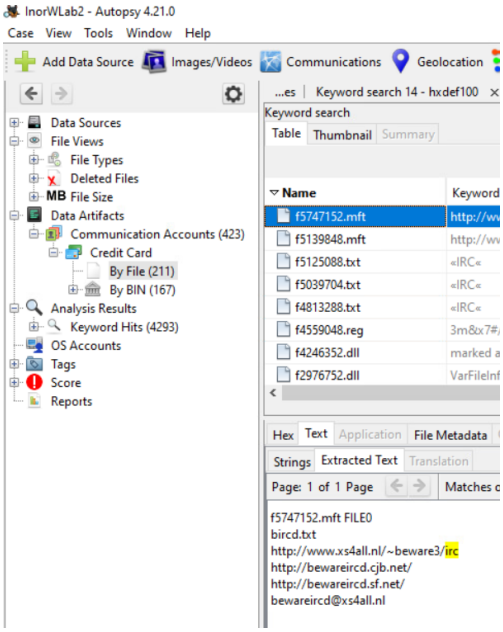

Figure 42: IRC autopsy search

I also noticed that there was another file in Autopsy which showed the settings of the IRC server. It's noted that it says that there's an admin of "Allen" and has a home network IP address of "192.168" and password of "JmzNRUUHpAYh2" as shown in Figure 42. This is an amazing finding because we can begin to start keyword searching the name, "Allen". Also, it states that the headline of the IRC server is "Welcome to the localhost IRC server. Where all the latest gamez, appz, and moviez are." I am assuming that Allen hacked into Daniel Faraday's computer system to host a IRC server to sell pirated games, apps, and movies. This is because whenever you search "hxdef100" within Autopsy,

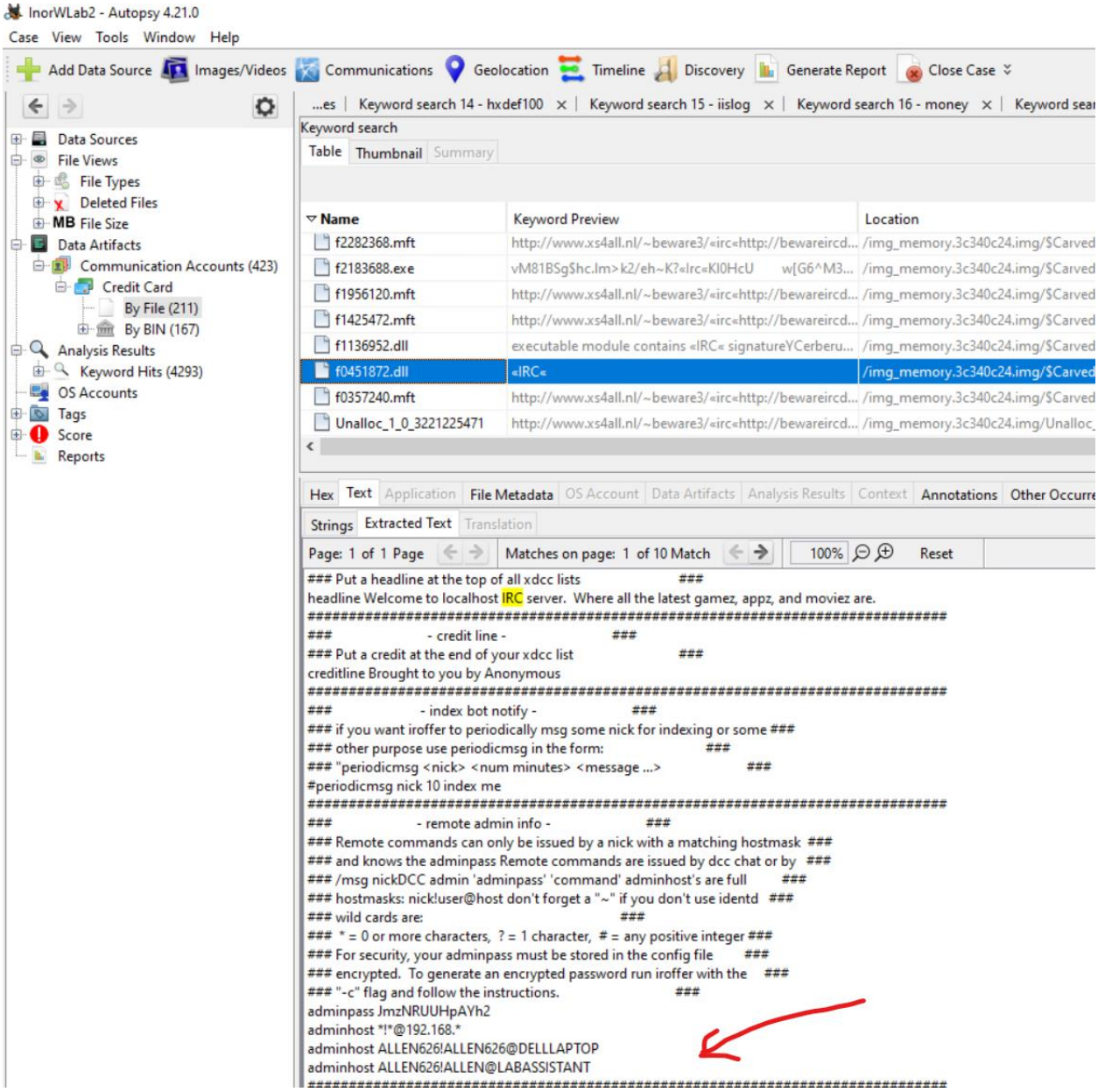

Figure 43: Found the hacker

I still had a question in my head, lingering. How did the hacker change the password of Daniel Faraday without having admin privileges nor knowledge to his password? Within Autopsy, I searched the common password cracking tool, “John The Ripper” as shown in Figures 44 and 45. Within Figure 44, it shows that Allen downloaded not only John the Ripper but also a back door rootkit client. As noted before, Allen set up that backdoor on port 6666 and this further solidifies our analysis of that. Within Figure 45, it shows John The Ripper.zip file but it also shows multiple other games, apps, and movies that are not only used to sell but also used to control the machine. Also as shown in Figure 46, it shows that mkpasswd.exe was used and deleted which is super suspicious. It also also suspicious knowing that John the Ripper was used within this system so finding that executable file alongside suspicious activity strongly hints at password

manipulation/cracking and is how Allen was able to gain elevated access and change Daniel Faraday's password. Also as noted in Figure 5, there was an open port 666 of Doom which is a video game. Which further solidifies that the hacker was most likely selling video games, apps, and movies, but also playing video games as well.

```
ALLEN626!ALLEN@LABASSISTANT;
chdesc 2 Download at : ://www.openwall.com/john/
dJhA
ALLEN626!ALLEN@LABASSISTANTK
chdesc 3 Download at : http://www.foofus.net/fizzgig/fgdump/
~}s!
C:/hidden/ir/packs/bdcli100.exe
bdcli100.exe : Back door rootkit client : For Host enter computer's IP. Use port 80. No password (just hit enter)
C:/hidden/ir/mybot.pid
C:/hidden/ir/mybot.log
logs+
C:/hidden/ir/mybot.state
626!
ahost
```

Figure 44: Showing that Allen downloaded John the Ripper and a Back door rootkit client

```
INDX(           :
bdcli100.exe
Fallout.3.USA.PROPER.RETAIL.XBOX360-x360inT.rarsE
FALLOU~1.RAR
John The Ripper.ziptE
JOHNTH~1.ZIP
Poison Ivy.rar
POISON~1.RAR
pwdump.tar.bz2
PWDUMP~1.BZ2
r0-f3apx.rar
The.Hurt.Locker.2008.720p.BluRay.x264-CiRCLE.rar
The.Sims.3.Razor.1911.rar
THEHUR~1.RAR
THESIM~1.RAR
X-Men.Origins.Wolverine.REAL.PROPER.WORKPRINT.XviD-iLG.rar
X-MENO~1.RAR
Actx
```

Figure 45: Showing John the Ripper.zip file and multiple other games and movies

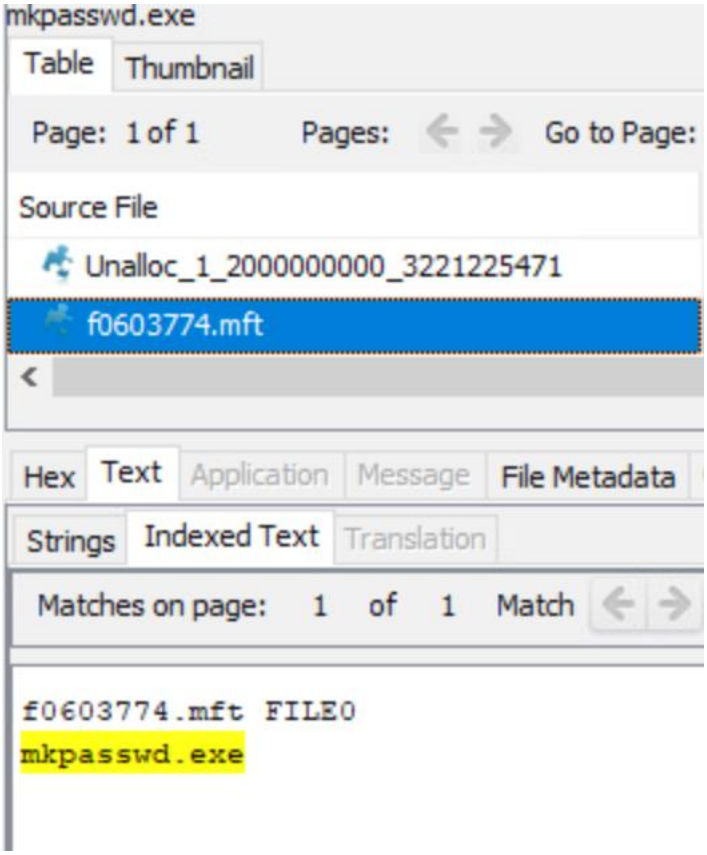

Figure 46: mkpasswd.exe file

Story

Allen first gained access to Daniel Faraday's Windows XP system by exploiting the built-in FTP server that allowed anonymous logins and lacked basic security configurations. From there, the FTP logs (in C:\Inetpub\ftproot and C:\WINDOWS\system32\Logfiles\MSFTPSVC1) showed that Allen uploaded a variety of tools, most notably Netcat (nc.exe), a VNC server (winvnc4.exe), runasspc.exe, and batch scripts designed to automate malicious actions. Immediately after positioning these files, he used John the Ripper (as evidenced by references to "John the Ripper.zip" and mkpasswd.exe in the logs and Autopsy artifacts) to crack Daniel Faraday's account credentials. He changed the password to "Bond007" in the registry (under HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon), ensuring his continued access even if administrative controls were later applied. Once inside, Allen locked out other users by running a simple script, lock.bat, which invoked rundll32.exe user32.dll, LockWorkStation. This forced anyone physically present at the Windows XP machine back to the login screen. To keep the system in a perpetually compromised state, Allen placed an additional batch file, startup.bat, in the Startup directory (C:\Documents and Settings\Daniel Faraday\Startup). This script automatically launched nc.exe on port 6666, ran the VNC server (winvnc4.exe), and then executed lock.bat again, so with every reboot, Allen's backdoor on port 6666 was re-established, a remote desktop session became available, and the legitimate user was repeatedly locked out. In parallel, Allen installed multiple stealth and persistence tools. The presence of hxdef100.exe in the C:\hxdefrootkit\ directory (and the associated crash logs from Dr. Watson in the Event Viewer) confirmed that he had deployed Hacker Defender, a well-known user-mode rootkit, to hide his processes and files. Prefetch files (e.g., POISONIVY.EXE-0ED7FCCD.pf) in C:\WINDOWS\Prefetch indicated Allen actually ran Poison Ivy, a powerful Remote Access Trojan capable of keylogging, file transfers, and more. Another Prefetch file showed cryptcat.exe had been launched, pointing to encrypted network sessions. The cryptcat tool would have allowed Allen to communicate securely with remote servers, bypassing standard network detection methods. Meanwhile, the bircd.exe (an IRC server executable) and configuration snippets discovered in Autopsy revealed Allen was running a personal IRC server, advertised under "Welcome to the localhost IRC server. Where all the latest gamez, appz, and moviez are.", likely to distribute pirated software or coordinate with co-conspirators in real time. Further digging in the FTP directories uncovered a file named Razor.1911.IRC, a README describing cracked copies of The Sims 3 and referencing other pirated titles. Traces of Doom (listening on port 666) and multiple MSN Gaming Zone directories also appeared on the machine, suggesting Allen was both playing and potentially hosting illicit or cracked games. Autopsy searches and Redline memory analysis corroborated these findings, illustrating a broad pattern of unauthorized software installation, malicious activity, and file transfers. IIS web server logs (W3SVC1) added another layer to Allen's multi-pronged compromise, showing that he (and possibly other attackers) probed default pages like iisstart.asp, confirming the operating system's weak internal security. Overall, Allen's chain of attack was multifaceted:

1. Initial Entry and Tool Upload: Exploited anonymous FTP to transfer nc.exe, runasspc.exe, VNC, and other utilities.
2. Credential Attacks: Used John the Ripper and mkpasswd.exe to change the default user password to “Bond007,” confirmed in the Winlogon registry key.
3. Lockout: Employed lock.bat to repeatedly lock the workstation, blocking legitimate users from interrupting malicious activities.
4. Persistence: Placed startup.bat in Daniel Faraday’s Startup folder to automatically re-launch Netcat, VNC, and the lock script on boot.
5. Rootkit and RAT Usage: Deployed Hacker Defender (hxdef100.exe) to hide processes, and Poison Ivy (poisonivy.exe) for advanced Remote Access Trojan capabilities.
6. Stealth and Encryption: Used cryptcat.exe for encrypted communications, drastically reducing the chance of straightforward network detection.
7. Unauthorized Services: Hosted an IRC server (bircd.exe) for file sharing or coordinating attacks, and possibly ran a Doom server on port 666, indicating both entertainment and deeper distribution of pirated content.
8. Piracy and File-Sharing: Downloaded or served cracked software (Razor.1911.IRC, The Sims 3 updates, and other game directories) and stored them in C:\Inetpub\ftproot or hidden directories.

In short, Allen systematically took over Daniel Faraday’s Windows XP system for both personal use, like playing classic games, and for running a small underground market of pirated software and films. The combination of rootkits, password cracking, remote backdoors, and repeated locking of the legitimate user left the machine entirely under his control. This sequence of events was thoroughly documented through log analysis, Prefetch findings, Registry settings, memory forensics, and file system searches, offering clear evidence of every stage of Allen’s unauthorized intrusion and continued exploitation of the compromised Windows XP machine.

Conclusion

This lab really shows why cybersecurity is so important and what can happen when even small things, like an unpatched OS or an anonymous FTP login, get overlooked. By digging into a compromised Windows XP system, we got hands-on experience with the same forensic tools and methods that actual incident responders use. We also saw how easy it can be for attackers to take over a machine, hide rootkits, and keep backdoors open if no one’s paying attention to basic security steps like turning off anonymous FTP or keeping the OS updated. Beyond the technical skills, the lab highlighted why it's a big deal to document everything properly and keep track of evidence for both school projects and potential legal cases. Everything we ran into connects directly to modern threats, whether it's an old XP box or a brand-new setup. Learning how to spot malicious behavior, shut it down, and then explain it to others is a huge part of making sure we’re ready for whatever the cyber world throws at us next.

Bibliography

Bircd.org. (n.d.). https://www.bircd.org/

Chatgpt. (n.d.-a). https://chatgpt.com/

Claudiouzelac. (n.d.). *ROOTKIT.COM/HF/HXDEF100R/READMEEN.TXT at master · Claudiouzelac/rootkit.com*. GitHub. https://github.com/claudiouzelac/rootkit.com/blob/master/hf/hxdef100r/readmeen.txt

Cryptcat: Kali linux tools. Kali Linux. (2024, March 11). https://www.kali.org/tools/cryptcat/

Cyber.nj.gov. (n.d.). https://www.cyber.nj.gov/threat-landscape/malware/trojans/poison-ivy

Download VNC server by realvnc®. RealVNC®. (2024, May 23). https://www.realvnc.com/en/connect/download/vnc/?lai_vid=WKzB6Wdm8HBy8&lai_sr=5-9&lai_sl=l

Fgdump: Take *that* LSASS! (n.d.). http://foofus.net/goons/fizzgig/fgdump/

GeeksforGeeks. (2024, July 1). *Prefetch files in windows*. https://www.geeksforgeeks.org/prefetch-files-in-windows/

Help center. (n.d.). https://kb.printerlogic.com/s/article/What-is-the-difference-in-the-IIS-logs-W3SVC1-W3SVC2

int0x33. (n.d.). *INT0X33/NC.EXE: Netcat for windows 32/64 bit*. GitHub. https://github.com/int0x33/nc.exe/

John the ripper password cracker. Openwall. (n.d.). https://www.openwall.com/john/

Mkpasswd(1) - linux man page. (n.d.-b). https://linux.die.net/man/1/mkpasswd

Munoz, J. (n.d.). personal.

NCAT - Netcat for the 21st Century. (n.d.). https://nmap.org/ncat/

Registry: Hkey_local_machine\software\microsoft\windows nt\currentversion\winlogon. (n.d.). https://renenyffenegger.ch/notes/Windows/registry/tree/HKEY_LOCAL_MACHINE/Software/Microsoft/Windows-NT/CurrentVersion/Winlogon/index

Script to lock a workstation. Spiceworks Community. (2020, February 21). https://community.spiceworks.com/t/script-to-lock-a-workstation/752272

Wikimedia Foundation. (2024, December 18). *Netcat*. Wikipedia. https://en.wikipedia.org/wiki/Netcat

Wikimedia Foundation. (2025a, February 27). *IRC*. Wikipedia. https://en.wikipedia.org/wiki/IRC

Wikimedia Foundation. (2025b, March 4). *Tar (computing)*. Wikipedia. https://en.wikipedia.org/wiki/Tar_(computing)

Wikimedia Foundation. (2025c, March 7). *Rootkit*. Wikipedia. https://en.wikipedia.org/wiki/Rootkit

Wikimedia Foundation. (2025d, March 12). *Windows XP*. Wikipedia. https://en.wikipedia.org/wiki/Windows_XP

Wikimedia Foundation. (2025e, March 16). *Internet information services*. Wikipedia. https://en.wikipedia.org/wiki/Internet_Information_Services#:~:text=Microsoft%20IIS%20(Internet%20Information%20Services,coding%20optimization%2C%20sitemaps%20/%20robots.

Yasar, K. (2022, October 20). *What is a rat (Remote Access Trojan)?: Definition from TechTarget*. Search Security. https://www.techtarget.com/searchsecurity/definition/RAT-remote-access-Trojan

Yasar, K., & Lockhart, E. (2022, June 15). *What is the windows registry editor? - definition from techtarget.com*. Search Enterprise Desktop. https://www.techtarget.com/searchenterprisedesktop/definition/Windows-Registry-Editor

Zenmap - official cross-platform nmap security scanner gui. (n.d.). https://nmap.org/zenmap/