Table of Contents

Examining a Forensic Image with Autopsy 2
Network Forensic using Wireshark 4
Rhino Hunt with Autopsy 7
Rhino Hunt with Wireshark 9
Memory Analysis with Autopsy 12
Memory Forensics of LastPass and Keeper 15
Capturing and Examining the Registry 18
Examining a Windows Disk Image 20
Email Forensics: 23
Android Studio Emulator 26
Rooting Android Studio's Emulator 29
Forensic Acquisition from Android 32
Android Analysis with Autopsy 35
Making a Rooted Android Emulator 37
iPhone Analysis with Autopsy 40
Windows and Linux Machines 42
Velociraptor Server on Linux 44
Investigating a PUP with Velociraptor 48
Investigating a Bot with Velociraptor 51
Investigating a Two-Stage RAT with Velociraptor 55

Examining a Forensic Image with Autopsy

Summary:

This report delineates the procedures undertaken to scrutinize a forensic image utilizing Autopsy. The process encompassed the installation of Autopsy, resolution of common issues, retrieval of the evidence file, establishment of a case, and identification of specific flagged information within the digital image.

Introduction:

Autopsy serves as an open-source digital forensics platform utilized for scrutinizing hard drives, smartphones, and similar storage devices. In this instance, Autopsy was deployed to inspect a forensic image dubbed "F200.E01" in order to unearth evidence.

Case Information:

Case Name: F200

Case Number: F200

Evidence File: F200.E01

Tools Used:

Autopsy

Procedure:

1. Installing Autopsy:

Installed Autopsy in single user mode.

2. Downloading the Evidence File:

Downloaded the evidence file "F200.E01" on the Windows machine.

3. Creating a Case in Autopsy:

Launched Autopsy and created a new case named "F200".

Selected the Documents folder as the case location and assigned it the case number "F200".

Added the evidence file "F200.E01" to the case.

Configured the ingest options and completed the case setup process.

4. Investigation:

Explored the data Autopsy found.

Located a file containing a message starting with "The flag is".

Captured an image showing the flag for documentation purposes.

Artifacts and Analysis:

Task	Artifact	Relevance	Tools Used	Description	Evidence Found
Flag 1	File Containing Message	Indication of pertinent data	Autopsy	Examination of files for specific message content	The flag is “EVIDENCE”

Conclusion:

The examination of the forensic image "F200.E01" using Autopsy was successful. The specified flag was located within a file, providing evidence for. Autopsy proved to be an effective tool for digital forensics investigations.

Network Forensic using Wireshark

Summary

This report outlines the steps taken to perform network forensics utilizing Wireshark. The analysis encompassed the scrutiny of network traffic captured in pcapng files to reveal sensitive data such as FTP and HTTP passwords, recognize encrypted communication protocols, ascertain the encryption tool utilized, and probe into APT attacker traffic.

Introduction:

Wireshark serves as a robust network protocol analyzer employed for network diagnostics, examination, software, and communications protocol development, as well as educational purposes. In this scenario, Wireshark was employed to scrutinize captured network traffic, extract pertinent data, and pinpoint security weaknesses.

Tools Used:

Wireshark

Procedure:

1. Installing Wireshark:

- Installed Wireshark on the local machine.

2. Examining Layers 1-4:

- Downloaded the file "FTPlogin.pcapng" and opened it in Wireshark.
- Analyzed the packet list, packet details, and packet bytes to understand the TCP handshake, MAC addresses, IP addresses, and TCP port numbers.
- Filtered FTP packets to locate FTP login processes and extract passwords.

3. Finding an FTP Password:

- Filtered packets using the FTP protocol.
- Identified the login process for a user named "john".
- Extracted John's password from the captured packets.

4. Finding an HTTP Password:

- Downloaded the file "httplogin.pcapng" and opened it in Wireshark.
- Filtered packets using the HTTP protocol.

- Identified HTTP POST requests to extract usernames and passwords.

5. Following a TCP Stream:

- Clicked on the first POST request and followed the TCP stream to examine client-server communication.
- Attempted to locate passwords in the TCP stream.

6. Restoring the Packet Filter:

- Cleared the filter to view all HTTP packets.
- Located and examined HTTP replies to determine login success or failure.

7. Analyzing APT Capture:

- Downloaded the file "apt-capture.pcap" containing APT attacker traffic.
- Identified encrypted communication protocols.
- Determined the port number and tool used for encryption.
- Calculated the sha1sum of the tool.
- Decrypted communication to find the private key and extract the flag.
- Analyzed port knocking patterns and identified the sequence of ports.
- Investigated shell sessions to identify the largest session and extract the last command executed.

Artifacts and Analysis:

Task	Artifact	Relevance	Tools Used	Description	Evidence Found
Finding FTP password	FTP Packets	Revealing insecure passwords	Wireshark	Analyzed FTP Packets to extract user passwords	John's Password is "Flapper".
Finding HTTP password	HTTP Packets	Revealing insecure passwords	Wireshark	Analyzed HTTP Packets to extract user passwords	Isaac's Password is "Flapper". Waldo's password is "VERYSCURE".

Encrypted Transmissions	Encrypted Packets	Identifying encryption protocols	Wireshark	Analyzed encrypted packets to determine protocol	Server Port Number is 8443.
Tool Identification	Encrypted Packets	Identifying encryption tool	Wireshark	Determined the tool used for encryption	“Socat”
Hash Of Tool	Encrypted Packets	Verifying encryption tool	Wireshark	Calculated Sha1 sum of encrypted tool	Hash value = “da8ff40c6a60605c4d250aae59912f1e60315c81”
Decrypting Communication	Encrypted Packets	Decryption Of Communication	Wireshark	Decrypted Communication to extract sensitive data	NCX{3e446ab1-9093-47eb-a59a-cdd49c28e0e0}
Port - Knocking	Network Traffic	Identifying port knocking	Wireshark	Analyzed port knocking patterns	32001,33001,34001
Exploitation	Shell Session	Identifying shell exploitation	Wireshark	Investigated shell sessions for exploitation	Stream ID = 75

Conclusion:

The examination of network traffic using Wireshark revealed significant security vulnerabilities, including insecure transmission of FTP and HTTP passwords, encrypted communication, port knocking activities, and shell exploitation attempts. By leveraging Wireshark's capabilities, sensitive information was extracted, and potential security threats were identified, highlighting the importance of network forensics in cybersecurity investigations.

Rhino Hunt with Autopsy

Summary

This report outlines the procedures followed to investigate a case involving the possession of illegal rhinoceros images using Autopsy. The investigation included verifying hash values, examining a disk image, identifying specific files, reading a diary, locating a missing hard drive, and identifying an email address associated with MIT.

Introduction:

The case involves the possession of illegal rhinoceros' images, which is a serious crime according to the law passed in the city of New Orleans in 2004. The evidence includes a disk image of a USB key and network traces provided by the network administrator. Autopsy, a digital forensics tool, was used to analyze the evidence and extract pertinent information.

Tools Used:

- Autopsy
- PowerShell

Procedure:

1. Verifying Hash Values:

- Opened a PowerShell window and executed commands to calculate MD5 and SHA1 hash values of the evidence file "case1.zip".
- Verified that the hash values matched the expected values to ensure data integrity.

2. Unzipping the Evidence File:

- Extracted the contents of the "case1.zip" file to access the disk image and network traces.

3. Creating an Autopsy Case:

- Launched Autopsy and created a new case named "F201".
- Added the disk image "RHINOUSB.dd" as a data source to the case.
- Configured ingestion settings and completed the data source setup process.

4. Investigating the Evidence:

- Explored the containers in Autopsy to view images and deleted files.

- Located the image of a mother rhinoceros and her child as per the provided flag.
- Examined deleted files and sorted them by MIME type to prioritize "application/msword" files.
- Read the diary file to uncover the flag.
- Identified the location of the missing hard drive within the disk image.
- Identified the real filename containing an email address associated with MIT as the flag.

Artifacts and Analysis:

Task	Artifact	Relevance	Tools Used	Description	Evidence Found
Verifying Hash Values	MD5 and SHA1 hash values	Ensuring data integrity	PowerShell	Calculated hash values to verify file authenticity	Hash values were matched
Investigating Deleted Files	Deleted files	Identifying potentially relevant data	Autopsy	Examined deleted files for valuable information	Found various deleted files including pictures of rhinoceros and crocodiles.
Reading the Diary	Diary file	Extracting information	Autopsy	Read diary to uncover pertinent information	Found location of hard drive and e-mail address
Locating Missing Hard Drive	Disk image analysis	Identifying specific location	Autopsy	Identified location of missing hard drive	The location of missing hard drive is “Mississippi river”.
Email Address Identification	Recovered files	Identifying email address	Autopsy	Identified real filename containing email address	Real filename is “f0103512.jpg”.

Conclusion:

The investigation using Autopsy revealed crucial information related to the possession of illegal rhinoceros images. Various artifacts were analyzed, including deleted files, diary entries, and recovered emails. The findings provide valuable insights into the case, facilitating further investigation and potential legal action.

Rhino Hunt with Wireshark

Summary

This report outlines the procedures followed to conduct network forensics using Wireshark in the Rhino scenario. The investigation involved analyzing network traffic captured in log files to identify TELNET and FTP packets, examining TELNET and FTP traffic streams, extracting images from FTP-DATA traffic, and verifying hash values of extracted files.

Introduction:

The Rhino scenario involves investigating illegal rhinoceros images-related activities using network traffic captured in log files. Wireshark, a network protocol analyzer, was utilized to analyze the network traffic and extract valuable information such as filenames, transferred files, and image data.

Tools Used:

- Wireshark
- PowerShell

Procedure:

1. Opening Rhino.log:

- Opened the "rhino.log" file in Wireshark to analyze the captured network traffic.

2. Finding TELNET Packets:

- Filtered the network traffic for TELNET packets and exported the displayed packets to a new file named "telnet.pcap".

3. Finding FTP Packets:

- Filtered the network traffic for FTP packets and exported the displayed packets to a new file named "ftp.pcap".

4. Examining TELNET Traffic:

- Closed the current file in Wireshark and opened the "telnet.pcap" file.
- Followed the TCP stream of the first displayed packet to analyze TELNET traffic and uncover relevant information.
- Identified the filename left as a message to John, which serves as the flag.

5. Examining FTP Traffic:

- Followed the same process as TELNET traffic to analyze FTP traffic.
- Identified the filename of the third transferred file, which serves as the flag.

6. Extracting Images from FTP-DATA Traffic:

- Loaded the original "rhino.log" file in Wireshark and extracted FTP-DATA packets to a separate file.
- Analyzed the TCP stream to identify JPG image data and saved it as "rhino1.jpg".
- Verified the integrity of the extracted image by calculating its MD5 hash.

7. Extracting ZIP Archive from FTP-DATA Traffic:

- Identified and extracted the ZIP archive from FTP-DATA streams indicated by the PK header.
- Unzipped the archive using an online tool to obtain the password-protected image file.
- Calculated the MD5 hash of the extracted image to verify its integrity and uncover the flag.

Artifacts and Analysis:

Task	Artifact	Relevance	Tools Used	Description	Evidence Found
Analyzing TELNET Traffic	TELNET packets	Identifying user actions	Wireshark	Analyzed TELNET traffic for relevant information	Filename for message to John is “JOHNREADME”.
Analyzing FTP Traffic	FTP packets	Identifying transferred files	Wireshark	Analyzed FTP traffic for transferred filenames	Filename of third transferred file is “contraband.zip”
Extracting Images from FTP-DATA	FTP-DATA packets	Identifying image data	Wireshark	Extracted images from FTP-DATA traffic	Rhino1.jpg
Verifying MD5 Hash of Images	Extracted image files	Ensuring data integrity	PowerShell	Verified integrity of extracted image files	Hash values are same.

Conclusion:

The investigation using Wireshark in the Rhino scenario revealed valuable information related to illegal rhinoceros images-related activities. TELNET and FTP traffic were analyzed to identify filenames and transferred files, while image data was extracted from FTP-DATA traffic. The integrity of extracted files was verified using MD5 hash values, ensuring the reliability of the findings.

Memory Analysis with Autopsy

Summary

This report details the procedures followed to extract useful information from a memory image using Autopsy. The investigation involved installing 7-Zip, verifying the integrity of the memory image, launching Autopsy, enabling the Experimental module, creating a new case, importing the memory image, and analyzing various artifacts to uncover flags related to passwords, hashes, executables, and more.

Introduction:

The purpose of this project was to conduct digital forensics analysis on a memory image to extract valuable information such as passwords, hashes, executable names, and other artifacts. Autopsy, along with additional tools such as 7-Zip and Hashcalc, were utilized to perform the analysis.

Tools Used:

- Autopsy
- 7-Zip
- Online Hash Calculator

Procedure:

1. Installing 7-Zip:

- Installed 7-Zip by downloading the 64-bit version from the official website and running the installer.

2. Downloading the Evidence File:

- Downloaded the memory image file "memdump.7z" and extracted it using 7-Zip to obtain the file "memdump.mem".
- Verified the hash of the extracted file using Online Hash Calculator to ensure data integrity.

3. Launching Autopsy:

- Launched Autopsy and enabled the Experimental module from the Plugins menu.

4. Creating a New Case:

- Created a new case named "memory" in Autopsy.

5. Importing the Memory Image:

- Added the memory image file "memdump.mem" as a data source to the case.
- Selected the appropriate data source type as "Memory Image File (Volatility)".
- Configured the ingest process by selecting specific plugins for analysis to speed up the process if required.
- Analyzed the output in the ModuleOutput section to uncover flags related to passwords, hashes, executables, and other artifacts.

Artifacts and Analysis:

Task	Artifact	Relevance	Tools Used	Description	Evidence Found
Consoles	Consoles section	Identifying user passwords	Autopsy	Located Waldo's password in console logs	Waldo's password is “Apple123”
Hash dump	Hash dump section	Identifying user password hashes	Autopsy	Located Waldo's password hash in hashdump logs	Waldo's password hash is “cfeac129dc5e61b2eb9b27131fc7e2b”
LSA dump	LSA dump section	Identifying default password	Autopsy	Identified default password in LSAdump logs	Default password is “P@ssw0rd”
Netscan	Netscan section	Identifying listening executable	Autopsy	Identified executable listening on port 8080	Executable name is “ftpbasicsvr.exe”
Pslist	Pslist section	Identifying running executables	Autopsy	Identified running executable from pslist logs	Executable name is “FTK Imager.exe”
Shell bags	Shell bags section	Identifying connected shared folder	Autopsy	Identified connected shared folder name	Shared folder name is “sambowne”
User assist	User assist section	Identifying dangerous executable	Autopsy	Identified dangerous executable from userassist logs	Executable name is “Poison Ivy 2.3.2.exe”

Probe Password	Hash dump section	Identifying probe account password	Autopsy	Located probe account password hash	Probe account password is “P@ssw0rd”.

Conclusion:

The investigation using Autopsy on the memory image provided valuable insights into user activities, passwords, hashes, executables, and other artifacts. By analyzing various sections and logs within Autopsy, multiple flags were uncovered, providing crucial information for further analysis and potential security measures.

Memory Forensics of LastPass and Keeper

Summary

This report documents the analysis conducted on LastPass and Keeper, two popular password managers, to understand how they handle sensitive user data in memory. The investigation involves installing necessary tools, setting up LastPass and Keeper accounts, exploring memory processes associated with their browser extensions, and searching for sensitive data strings in memory using Process Explorer and HxD.

Introduction:

The purpose of this project was to examine the behavior of LastPass and Keeper password managers regarding the storage and handling of sensitive user data, specifically passwords, in the system memory. By installing the respective browser extensions and analyzing memory processes, the aim was to identify potential security risks associated with these password managers.

Tools Used:

- Google Chrome
- Mozilla Firefox
- Process Explorer
- HxD Hex Editor

Procedure:

1. Installing Required Software:

- Installed Google Chrome, Mozilla Firefox, Process Explorer, and HxD Hex Editor on the Windows virtual machine.

2. Target 1: LastPass

Installing LastPass:

- Installed LastPass browser extension in Google Chrome.
- Created a LastPass account with a disposable email address.
- Added a fake password entry for testing purposes.

Exploring LastPass Memory Process:

- Launched Process Explorer and identified the Chrome process containing the LastPass extension using the "--extension-process" switch.
- Identified the Process ID (PID) of the LastPass extension process.
- Launched HxD and opened the main memory of the LastPass extension process.
- Searched for the test password string "testpassword1234" in Unicode format.

3. Target 2: Keeper

Installing Keeper:

- Installed Keeper browser extension in Mozilla Firefox.
- Created a Keeper account with a disposable email address.
- Added a login record with a password containing the string "CCSF#".

Finding Keeper Memory Process:

- Explored Firefox processes in Process Explorer.
- Looked for clues such as DLL names containing "keepersecurity" or "formautofill", or handles to File \Device\KsecDD.
- Searched for the string "CCSF#" in Unicode format within the identified Firefox process memory using HxD.

Artifact Analysis:

Task	Artifact	Relevance	Tools Used	Description	Evidence Found
LastPass Memory Analysis	Memory process	Identifying stored password	Process Explorer, HxD	Searched for test password string "testpassword1234"	The flag is “Excerpt”
Keeper Memory Analysis	Memory process	Identifying stored password	Process Explorer, HxD	Searched for string "CCSF#" in memory process	The flag is “43 00 43 00”

Conclusion:

The analysis of LastPass and Keeper password managers revealed potential security risks associated with the handling of sensitive user data in memory. By examining memory processes, it was possible to identify instances where plaintext passwords were present, posing a threat to user security. Further investigation and monitoring of such password managers are necessary to ensure the protection of user data.

Capturing and Examining the Registry

Summary

This report presents the findings and analysis conducted in a digital forensics case study involving the examination of a memory image using Autopsy. The investigation aimed to extract volatile data from the registry image to uncover valuable evidence for legal proceedings. The report outlines the artifacts discovered, tools employed, acquisition process, analysis procedure, and evidence found during the investigation.

Introduction:

Digital forensics plays a crucial role in investigating and analyzing digital devices and data to uncover evidence for legal proceedings. In this report, we delve into a specific case study involving the analysis of a memory image with Autopsy. The investigation required a systematic approach to extract volatile data from the registry image and uncover relevant evidence.

Tools Used:

- Autopsy
- FTK Imager
- Registry Editor

Procedure:

- Obtain the registry image using appropriate tools and techniques, ensuring the preservation of integrity.
- Utilize Autopsy to examine memory artifacts and extract relevant information.
- Document all findings, including artifacts discovered, analysis procedures followed, and evidence obtained.

Artifacts and Analysis:

Task	Artifact	Relevance	Tools Used	Description	Evidence Found
Examining the Registry Image with Autopsy	Registry Image	Contains volatile data from registry.	Autopsy, FTK Imager	Imported registry image into Autopsy. Analyzed registry artifacts using Autopsy plugins.	NTUSER.DAT file

Conclusion:

The digital forensics investigation conducted in this case study highlights the importance of employing specialized tools and methodologies to analyze memory images effectively. By leveraging Autopsy's capabilities, forensic examiners can extract valuable evidence from volatile data sources, aiding in legal proceedings and investigations.

Examining a Windows Disk Image

Summary

This report details the procedures followed to examine a forensic image obtained from the NIST Data Leakage Case. The investigation involved downloading evidence files from a provided URL, verifying the hash value, analyzing the evidence using Autopsy, and identifying specific artifacts to answer questions related to the case.

Introduction:

The NIST Data Leakage Case presents a scenario where forensic analysis is required to uncover potential evidence of data leakage. This report documents the steps taken to conduct digital forensics analysis using Autopsy, a widely used open-source forensic tool.

Case Information:

- Case Name: NIST Data Leakage Case
- Case Number: F221
- Evidence Files: cfreds_2015_data_leakage_pc.7z.001, cfreds_2015_data_leakage_pc.7z.002, cfreds_2015_data_leakage_pc.7z.003

Tools Used:

- 7-Zip
- Autopsy

Procedure:

1. Downloading the Evidence Files:

- Accessed the provided URL and downloaded the first three files ending in ".001", ".002", and ".003".

2. Installing 7-Zip:

- Visited the 7-Zip website and downloaded the installation package.
- Installed 7-Zip on the Windows machine.

3. Unzipping the Evidence File:

- Extracted the evidence file "cfreds_2015_data_leakage_pc.dd" using 7-Zip.
- Verified the extraction by confirming the file size.

4. Verifying the Hash (MD5):

- Opened a PowerShell window and navigated to the Downloads folder.
- Executed the command **Get-FileHash -Algorithm MD5 cfreds_2015_data_leakage_pc.dd** to obtain the MD5 hash value.
- Compared the obtained MD5 hash value with the expected value to ensure data integrity.

5. Analyzing the Evidence with Autopsy:

- Launched Autopsy and created a new case named "F221".
- Imported the evidence image "cfreds_2015_data_leakage_pc.dd" into the case.
- Configured ingest options and processed the data.

6. Data Artifacts Analysis:

- Explored various artifacts within Autopsy to answer specific questions related to the case.

Artifacts and Analysis:

Task	Artifact	Relevance	Tools Used	Description	Evidence Found
Verifying the Hash (MD5)	cfreds_2015_data_leakage_pc.dd	Data integrity verification	PowerShell	Calculated MD5 hash value to ensure file integrity	Hash values were same
Most Recently Installed Program	Installed Programs	Identification of recent software installations	Autopsy	Examined installed programs to identify the most recent installation	Eraser 6.2.0.2962 v.6.2.2962
Most Recent Document	Recent Documents	Identification of recent document access	Autopsy	Reviewed recent document access to determine the most recent document accessed	Resignation letter (laman_informant).xps.lnk
59 Times	Prefetch folder	Analysis of frequently run programs	Autopsy	Examined Prefetch folder to identify programs run 59 times	DLLHOST.EXE

Search	Web Search History	Identification of suspicious web searches	Autopsy	Investigated web search history to identify suspicious searches	Anti-forensic tools

Conclusion:

The digital forensics investigation of the NIST Data Leakage Case using Autopsy and 7-Zip was conducted successfully. The procedures outlined in this report ensured the integrity of the evidence files and facilitated the analysis of various artifacts to answer specific questions related to the case. Autopsy proved to be a valuable tool for examining digital evidence and uncovering potential evidence of data leakage.

Email Forensics:

Summary

This report outlines the digital forensic investigation conducted to identify the perpetrator behind harassing emails received by Lily Tuckrige, a teacher at Nitroba, suspected to be sent by a student in her Chemistry 109 class. The investigation involved analyzing email headers, capturing network traffic, and examining HTTP requests to determine the origin of the emails and potential suspects.

Introduction:

The Nitroba IT department received a complaint from Lily Tuckrige regarding harassing emails she received at her personal email account. The investigation aimed to trace the origin of the emails, identify the perpetrator, and determine if they were a student in Tuckrige's Chemistry 109 class.

Case Information:

- Case Name: Nitroba Harassment Case
- Case Number: NH001

Tools Used:

- Wireshark

Procedure:

1. Analyzing Email Headers:

- Obtained the full headers of the harassing email from Lily Tuckrige.
- Identified the IP address 140.247.62.34 as the origin of the email, located in a Nitroba student dorm room.

2. Capturing Network Traffic:

- Placed a network sniffer on the ethernet port of the dorm room.
- Logged all packets to analyze the traffic.

3. Analyzing HTTP Requests:

- Identified the use of "willselfdestruct.com" to send a harassing email.
- Extracted the IP address accessing "willselfdestruct.com" as 69.25.94.22.

- Traced the IP address 192.168.15.4 as the suspect's ID.

4. Examination of HTTP POST Requests:

- Searched for HTTP POST requests between 69.25.94.22 and 192.168.15.4.
- Found harassing comments in the HTTP POST requests.

5. Associating IP with Suspects:

- Identified MAC address 00:17:f2:e2:c0:ce associated with the suspect's Apple device.

6. Identification of Suspects:

- Narrowed down search results using cookies.
- Found potential suspects' profiles associated with cookies.
- Verified if the suspect was a student in Lily Tuckrige's Chemistry 109 class.

Artifacts and Analysis:

Task	Artifact	Relevance	Tools Used	Description	Evidence Found
Analyzing Email Headers	Email Headers	Trace email origin	Manual Analysis	Identified originating IP address 140.247.62.34	IP: 140.247.62.34
Capturing Network Traffic	Network Traffic Logs	Capture network activity	Network Sniffer	Logged all packets to analyze HTTP requests	Network Traffic Logs
Analyzing HTTP Requests	HTTP Requests	Trace website access	Wireshark	Extracted IP address 69.25.94.22 accessing "willselfdestruct.com"	IP: 69.25.94.22
Examination of HTTP POST Requests	HTTP POST Requests	Identify harassing comments	Wireshark	Found harassing comments in HTTP POST requests between 69.25.94.22 and 192.168.15.4	Harassing Comments
Associating IP with Suspects	MAC Address	Identify device	Network Sniffer	Associated MAC address 00:17:f2:e2:c0:ce	MAC Address

				with the suspect's Apple device	
Identification of Suspects	Cookies	Profile association	Wireshark	Found potential suspects' profiles	Originating email from Johnny Coach

Conclusion:

The digital forensic investigation successfully traced the origin of the harassing emails to a Nitroba student dorm room. By analyzing email headers, capturing network traffic, and examining HTTP requests, the perpetrator was identified using the web-based service "willselfdestruct.com." The suspect's IP address and MAC address were obtained, and potential suspects were narrowed down based on associated profiles. Further investigation is required to confirm the identity of the perpetrator and their association with Lily Tuckrige's Chemistry 109 class.

Android Studio Emulator

Summary

This report documents the process of setting up an Android emulator using Android Studio and installing a terminal emulator app called Qute for security auditing purposes. The steps involve downloading and installing Android Studio, creating a virtual Android device with Google Play support, troubleshooting common issues, and installing the Qute terminal emulator app.

Introduction:

The purpose of this project is to prepare an Android emulator environment for security auditing of Android applications. The setup involves configuring Android Studio, creating a virtual Android device, and installing a terminal emulator app to execute commands for auditing purposes.

Tools Used:

- Android Studio
- Qute: Terminal Emulator

Procedure:

1. Installing Android Studio:

- Downloaded Android Studio from the official website.
- Installed Android Studio with default options.
- Launched Android Studio and created a new project with a "Basic Views" activity.

2. Creating an Emulated Android Device:

- Accessed the Device Manager tab in Android Studio.
- Created a new virtual device using "Pixel 3a" hardware with Google Play support.
- Downloaded and installed the recommended system image.
- Configured advanced settings, including increasing internal storage.
- Launched the virtual device using the Run button.

3. Installing Qute: Terminal Emulator:

- Accessed Google Play from the virtual Android device.

- Logged in with a Google account.
- Searched for and installed "Qute: Terminal Emulator" from Google Play.

4. Executing Commands in Qute:

- Launched Qute: Terminal Emulator from the Android home screen.
- Agreed to the End User License Agreement (EULA) and Privacy Policy.
- Granted necessary permissions for the app.
- Executed the command **uname -a** to gather system information.

Artifacts and Analysis:

Task	Artifact	Relevance	Tools Used	Description	Evidence Found
Installing Android Studio	Android Studio Installation	Environment Setup	Manual Installation	Installed Android Studio with default options and created a new project	N/A
Creating an Emulated Android Device	Virtual Device Configuration	Emulation Setup	Android Studio	Created a virtual Android device with Google Play support	N/A
Installing Qute: Terminal Emulator	Qute Installation	Terminal Emulator Setup	Google Play	Installed "Qute: Terminal Emulator" app from Google Play	N/A
Executing Commands in Qute	Terminal Commands Execution	Auditing Environment Setup	Qute: Terminal Emulator	Executed the command **uname -a** in Qute terminal to gather	Flag is “linux”

				system information	

Conclusion:

The setup of the Android emulator environment for security auditing purposes was completed successfully. Android Studio was installed, a virtual Android device with Google Play support was created, and the Qute terminal emulator app was installed for executing commands. The Qute terminal emulator was used to gather system information, which provided valuable insights for security auditing.

Rooting Android Studio's Emulator

Summary

This report details the process of creating an Android emulator with root privileges using Android Studio. The objective is to set up a 32-bit x86 device with ARM translation libraries for maximum compatibility with old apps. The report outlines the steps involved in creating the emulator, adjusting RAM and storage sizes, starting the emulator, connecting with ADB, and opening a root shell to execute commands.

Introduction:

The purpose of this project is to create an Android emulator environment with root privileges to facilitate testing and debugging of applications that require elevated permissions. The emulator will be configured with specific hardware and system image settings to ensure compatibility with legacy applications.

Tools Used:

- Android Studio
- ADB (Android Debug Bridge)

Procedure:

1. Creating a Device:

- Launched Android Studio and accessed the Device Manager.
- Selected "Pixel 3a XL" hardware in the Select Hardware box.
- Chose "Android 11.0" without Google API in the System Image box and downloaded it.
- Adjusted RAM to 4096 MB and increased Internal Storage to 2048 MB in the Advanced Settings.
- Created the emulator device.

2. Start Your Emulator:

- Started the emulator from Android Studio.

3. Connecting with ADB:

- Executed the command **adb devices** in a Command Prompt.
- Ensured the emulator device is listed in the output.

4. Opening a Root Shell:

- Executed the commands **adb shell** and **su** to open a root shell in the Command Prompt.

5. Executing Commands for Flags:

- In the root shell, executed the command **ps** to find the line ending with "ps" and retrieved the flag.
- Executed the command **id** to retrieved the flag.

Artifacts and Analysis:

Task	Artifact	Relevance	Tools Used	Description	Evidence Found
Creating a Device	Emulator Configuration	Emulator Setup	Android Studio	Configured the emulator with specific hardware and system image settings	N/A
Start Your Emulator	Emulator Start	Emulator Initialization	Android Studio	Started the emulator from Android Studio	N/A
Connecting with ADB	ADB Connection	Device Connection	ADB	Connected to the emulator device using ADB	5554 Emulator Devices Listed
Opening a Root Shell	Root Shell	Privileged Access	ADB	Opened a root shell in the Command Prompt or Terminal	Root Shell
Executing Commands for Flags	Command Output	Task Completion	ADB	Executed commands **ps** and **id** in the root shell to retrieve flags for specific tasks	Ps = “root” Id = “su”

Conclusion:

The setup of an Android emulator with root privileges was successfully completed using Android Studio and ADB. The emulator was configured with appropriate settings, started, and connected to using ADB. A root shell was opened to execute commands, resulting in the retrieval of flags for specific tasks as part of the project requirements.

Forensic Acquisition from Android

Summary

This report outlines the process of collecting data from an Android emulator with root access and analyzing it using Autopsy. The steps involve starting the emulator, sending messages and making phone calls to generate user data, collecting the data from the emulator, transferring it to a Windows analysis machine, unzipping the data, and analyzing it with Autopsy.

Introduction:

The objective of this project is to demonstrate the collection and analysis of user data from an Android emulator environment. By simulating SMS exchanges and phone calls on the emulator, various types of user data are generated, including messages and call logs. The collected data is then analyzed using Autopsy to extract valuable insights for forensic examination.

Tools Used:

- Android Studio (for emulator setup)
- ADB (Android Debug Bridge)
- Autopsy
- 7-Zip

Procedure:

1. Starting the Emulator:

- Launched the Android emulator with root access prepared in a previous project.

2. Putting Evidence on the Phone:

- Followed the appropriate instructions for Windows users to send messages and make phone calls to the emulator.
- Opened Messages on the emulator, received and replied to the SMS.
- Answered the incoming call on the emulator.

3. Opening a Root Shell:

- Accessed a root shell on the emulator using ADB.
- Executed the command to collect user data from the /data directory.

4. Collecting User Data:

- Executed the command to create a tar archive of user data and save it to the emulator's storage.
- Transferred the tar archive to the Windows analysis machine using ADB.

5. Analyzing the Android Data with Autopsy:

- Launched Autopsy on the Windows analysis machine.
- Created a new case named "Android" in Autopsy.
- Imported the Android data as a logical file source.
- Configured the ingest options to use the Android Analyzer (aLEAPP).
- Examined the evidence in the Data Artifacts section of Autopsy.

Artifacts and Analysis:

Task	Artifact	Relevance	Tools Used	Description	Evidence Found
Starting the Emulator	Emulator Startup	Environment Initialization	Android Studio	Launched the Android emulator with root access	N/A
Putting Evidence on the Phone	SMS, Phone Call	User Interaction Data	ADB	Simulated SMS exchanges and phone calls on the emulator to generate user data	N/A
Opening a Root Shell	Root Shell	Privileged Access	ADB	Accessed a root shell on the emulator to collect user data	N/A
Collecting User Data	Tar Archive	Data Collection	ADB	Created a tar archive of user data from the emulator's /data directory	Tar Archive
Analyzing the Android Data with Autopsy	Autopsy Case, Data Artifacts	Forensic Analysis	Autopsy	Imported the collected data into Autopsy and examined the Data Artifacts section for analysis	Flag is “mmssms.db”

Conclusion:

The process of collecting data from an Android emulator with root access and analyzing it using Autopsy was completed successfully. SMS exchanges and phone calls were simulated to generate user data, which was then collected from the emulator and transferred to a Windows analysis machine. The collected data was analyzed using Autopsy, revealing valuable insights for forensic examination.

Android Analysis with Autopsy

Summary:

This report outlines the process of analyzing data from an Android device using Autopsy. The data, contained in the android_image2.tar.gz file, was extracted, imported into Autopsy, and examined to identify various artifacts such as phone calls, messages, and web searches. Specific tasks were performed within Autopsy to answer questions regarding the most recently installed app and the website viewed at a particular time.

Introduction:

The objective of this project is to analyze data extracted from an Android device using Autopsy, a digital forensics tool. The analysis aims to uncover valuable information such as the most recently installed app and the website viewed at a specific time, providing insights for forensic examination and investigation.

Tools Used:

- Autopsy
- 7-Zip
- Online Hash Calculator

Procedure:

1. Downloading the Evidence File:

- Downloaded the android_image2.tar.gz file from the provided link using a web browser.

2. Verifying the Hash Value:

- Calculated the hash value of the android_image2.tar.gz file using hash calculator.
- Verified that the calculated hash value matches the provided hash value.

3. Unzipping the Data:

- Deleted any existing "data" folder on the Windows desktop.
- Placed the android_image2.tar.gz file on the Windows desktop.
- Extracted the contents of the android_image2.tar.gz file using 7-Zip, resulting in a "data" folder.

4. Analyzing the Android Data with Autopsy:

- Launched Autopsy on the Windows analysis machine.
- Created a new case named "Android2" in Autopsy.
- Imported the Android data from the "data" folder as a logical file source.
- Configured the ingest options to use the Android Analyzer (aLEAPP).
- Examined the evidence in the Data Artifacts section of Autopsy.

Artifacts and Analysis:

Task	**Artifact**	**Relevance**	Tools Used	**Description**	**Evidence Found**
Most Recently Installed App	Installed Apps	User Activity	Autopsy	Identified the most recently installed app in the Android data	Com.u360mobile.usna
Website Viewed at Specific Time	Web Browsing History	User Activity	Autopsy	Determined the website viewed at 14:52:39 PDT on Oct 8, 2022	https://yahoo.com/

Conclusion:

The analysis of data from the Android device using Autopsy was successfully conducted. By importing the extracted data into Autopsy and examining the Data Artifacts section, valuable information regarding installed apps and web browsing activity was uncovered. The most recently installed app and the website viewed at a specific time were identified, providing significant insights for forensic investigation.

Making a Rooted Android Emulator

Summary

This report details the process of creating an Android emulator with root privileges using Android Studio. The objective is to set up a 32-bit x86 device with ARM translation libraries for maximum compatibility with old apps. The report outlines the steps involved in creating the emulator, adjusting RAM and storage sizes, starting the emulator, connecting with ADB, and opening a root shell to execute commands.

Introduction:

The purpose of this project is to create an Android emulator environment with root privileges to facilitate testing and debugging of applications that require elevated permissions. The emulator will be configured with specific hardware and system image settings to ensure compatibility with legacy applications.

Tools Used:

- Android Studio
- ADB (Android Debug Bridge)

Procedure:

1. Creating a Device:

- Launched Android Studio and accessed the Device Manager.
- Selected "Pixel 3a XL" hardware in the Select Hardware box.
- Chose "Android 11.0" without Google API in the System Image box and downloaded it.
- Adjusted RAM to 4096 MB and increased Internal Storage to 2048 MB in the Advanced Settings.
- Created the emulator device.

2. Start Your Emulator:

- Started the emulator from Android Studio.

3. Connecting with ADB:

- Executed the command **adb devices** in a Command Prompt.
- Ensured the emulator device is listed in the output.

4. Opening a Root Shell:

- Executed the commands **adb shell** and **su** to open a root shell in the Command Prompt.

5. Executing Commands for Flags:

- In the root shell, executed the command **ps** to find the line ending with "ps" and retrieved the flag.
- Executed the command **id** to retrieved the flag.

Artifacts and Analysis:

Task	Artifact	Relevance	Tools Used	Description	Evidence Found
Creating a Device	Emulator Configuration	Emulator Setup	Android Studio	Configured the emulator with specific hardware and system image settings	N/A
Start Your Emulator	Emulator Start	Emulator Initialization	Android Studio	Started the emulator from Android Studio	N/A
Connecting with ADB	ADB Connection	Device Connection	ADB	Connected to the emulator device using ADB	5554 Emulator Devices Listed
Opening a Root Shell	Root Shell	Privileged Access	ADB	Opened a root shell in the Command Prompt or Terminal	Root Shell
Executing Commands for Flags	Command Output	Task Completion	ADB	Executed commands **ps** and **id** in the root shell to retrieve flags for specific tasks	Ps = “root” Id = “su”

Conclusion:

The setup of an Android emulator with root privileges was successfully completed using Android Studio and ADB. The emulator was configured with appropriate settings, started,

and connected to using ADB. A root shell was opened to execute commands, resulting in the retrieval of flags for specific tasks as part of the project requirements.

iPhone Analysis with Autopsy

Summary

This report outlines the process of analyzing data from an iPhone using Autopsy. The data was obtained from the "2021 CTF - iOS.zip" file, which was extracted and imported into Autopsy for examination. Various artifacts such as phone calls, messages, and web searches were analyzed to answer specific questions related to the iPhone data.

Introduction:

The objective of this project is to conduct forensic analysis on data obtained from an iPhone using Autopsy, a digital forensics tool. The analysis aims to uncover relevant information such as phone numbers, GPS data, SMS messages, and program notifications, providing insights for forensic examination and investigation.

Tools Used:

- Autopsy
- 7-Zip
- PowerShell

Procedure:

1. Downloading the Evidence File:

- Accessed the webpage provided and downloaded the "2021 CTF - iOS.zip" file using a web browser.

2. Verifying the Hash Value:

- Calculated the SHA-256 hash of the "2021 CTF - iOS.zip" file using PowerShell commands.
- Verified that the calculated hash value matched the provided hash value.

3. Unzipping the Data:

- Extracted the contents of the "2021 CTF - iOS.zip" file using 7-Zip, resulting in a folder named "2021 CTF - iOS".

4. Analyzing the iPhone Data with Autopsy:

- Launched Autopsy on the Windows analysis machine.
- Created a new case named "iPhone" in Autopsy.

- Imported the iPhone data from the "2021 CTF - iOS" folder as a logical file source.
- Configured the ingest options to use the iOS Analyzer (aLEAPP).

Artifacts and Analysis:

Task	Artifact	Relevance	Description	Evidence Found
Phone Number	Device Paired via Bluetooth / Phone Number in 541 Area Code	Identification	Identified the device paired via Bluetooth or phone number in the 541 area code	Eli's Apple Watch / +15854178420
Latitude	GPS Last Known Location / Wireless Networks	Location Data	Determined the most northern latitude of the phone using GPS data or wireless network information	44.490257170
SMS	Messages	Communication	Identified the code sent by SIGNAL to the phone via SMS messages	191-116
Signal Contact	Program Notifications	Communication	Identified the person named Johnathan who sent Snapchat (pikaboo) messages to the phone	Jonathan Chipps

Conclusion:

The analysis of data from the iPhone using Autopsy was successfully completed. By importing the extracted data into Autopsy and examining the Data Artifacts section, various pieces of information such as phone numbers, GPS data, SMS messages, and program notifications were uncovered. This information provides valuable insights for forensic examination and investigation.

Windows and Linux Machines

Summary

This document provides step-by-step instructions for setting up two virtual machines (VMs) on Microsoft Azure: one running Windows 11 and the other running Debian 11 Linux. The process involves creating an Azure for Students account, provisioning the VMs, and configuring them for use in projects.

Introduction:

Setting up virtual machines on Azure allows for easy access to Windows and Linux environments for various projects and tasks. This guide outlines the process of creating and configuring VMs on Azure's cloud computing platform.

Prerequisites:

- An .edu email address
- Access to a phone for verification
- Web browser

Procedure:

1. Creating an Account:

- Visit the Azure for Students FAQ page.
- Click on the "Activate" button and follow the on-screen instructions to create an Azure for Students account using your .edu email address and phone number.

2. Creating a Windows 11 Cloud Server:

- Navigate to the Azure Education Hub and click on the three-bar "hamburger" icon.
- Select "Virtual Machines" and click on "Create Azure virtual machine".
- Fill in the required information on the Basics tab, including subscription, resource group, virtual machine name, region, image (Windows 11 Pro), and VM architecture.
- Enter additional details such as username, password, and licensing options.
- Review the configuration and click on the "Review + create" button.
- Follow any prompts or error messages to ensure successful creation.

- Once deployed, connect to the Windows 11 server using the provided IP address and appropriate client.
- Set up auto-shutdown to avoid unnecessary charges.

3. Creating a Debian 11 Linux Server:

- Follow similar steps as above but select the Debian 11 image and configure the VM accordingly.
- Download the private key file generated during deployment for SSH access.
- Connect to the Debian 11 server using SSH with the provided private key.

4. Recording Success:

- On the Windows machine, open Command Prompt and execute **dir c:\windows\system32\drivers\etc** to find the Windows flag. Flag = “hosts”.
- On the Linux machine, open an SSH window and execute **lsb_release -a** to find the Linux flag. Flag = “Debian”

Conclusion:

By following the steps outlined in this guide, two virtual machines running Windows 11 and Debian 11 Linux have been successfully provisioned on Azure. These VMs can now be used for various projects and tasks, providing flexibility and scalability in cloud computing environments.

Velociraptor Server on Linux

Summary

This report details the process of installing and configuring Velociraptor on a Debian Linux server to monitor a Windows endpoint. It includes steps for preparing the server, installing Velociraptor, configuring the server, creating an administrator user, starting the Velociraptor server, and accessing the GUI. Additionally, it covers the process of adding a Windows client, preparing a client configuration file, creating a client installer, transferring the installer to a Windows machine, installing the client, and troubleshooting potential issues.

Introduction:

Velociraptor is a digital forensics and incident response tool used for endpoint monitoring and analysis. This report documents the steps involved in setting up Velociraptor on a Linux server and configuring it to monitor a Windows endpoint. The process includes preparing the server, installing Velociraptor, configuring server settings, creating an administrator user, starting the server, accessing the GUI, adding a Windows client, preparing a client configuration file, creating a client installer, installing the client on a Windows machine, and troubleshooting common issues.

Case Information:

- Server OS: Debian Linux
- Client OS: Windows

Tools Used:

- Web browser
- Terminal or SSH
- Velociraptor

Procedure:

1. Installing Velociraptor on Linux:

- **Finding the Latest Version:**
    - Accessed the Velociraptor releases page on GitHub to identify the latest version.
    - Downloaded the 64-bit Linux installer and the Windows 64-bit EXE installer.

- **Preparing the Server:**
    - Created a directory for Velociraptor and downloaded the Linux installer.
    - Made the installer executable and generated a server configuration file.
- **Editing the Config File:**
    - Modified the server configuration file to replace localhost with the server's IP address.
- **Creating the Administrator User:**
    - Added an admin user with administrator role to the Velociraptor server.
- **Starting the Velociraptor Server:**
    - Initiated the Velociraptor server with the configured settings.

2. Adding a Windows Client:

- **Preparing a Client Config File:**
    - Edited the server configuration file to enable self-signed SSL.
- **Preparing a Windows Client Installer:**
    - Generated a client configuration file and downloaded the Windows client installer.
    - Repackaged the client installer with the configuration file.
- **Moving Client to the Windows Machine:**
    - Transferred the repackaged client installer to the Windows machine using WinSCP.
- **Installing the Windows Client:**
    - Executed the client installer on the Windows machine to install the Velociraptor client.

Artifacts and Analysis:

Task	Artifact	Relevance	Tools Used	Description	Evidence Found
Velociraptor Installation	Linux server configuration	Establishing server setup	Web browser, Terminal	Installed and configured Velociraptor on	Successful setup

				Debian Linux server	
	Windows client installer	Facilitating client installation	Web browser, Terminal	Repackaged Windows client installer with necessary configurations for monitoring Windows endpoint	Installer created
Windows Client Addition	Client configuration file	Configuring client settings	Terminal	Edited client configuration file to specify server IP address	Configuration done
	Windows client installer	Enabling Velociraptor monitoring	Terminal, WinSCP	Installed Velociraptor client on Windows machine for endpoint monitoring	Client installed
Server Name	GUI Home Page	Identifying server status	velociraptor	Located and identified flag on the home page of the Velociraptor GUI	“velociraptor”
Agent Name	Velociraptor Client	Identifying client status	velociraptor	Located and identified flag on the Velociraptor client page in the GUI	“Labels”
Registry Information	Velociraptor GUI	Accessing registry information	velociraptor	Retrieved registry information using Velociraptor GUI	“Keyboard”
Exploring the File System	Velociraptor GUI	Navigating file system	velociraptor	Explored file system using Velociraptor GUI	“NTUSER.DAT”
DestPort	Velociraptor GUI	Identifying destination ports	velociraptor	Retrieved destination port information using Velociraptor GUI	“8000”

Conclusion:

The installation and configuration of Velociraptor on a Debian Linux server and Windows client were successfully completed. The Velociraptor server was configured and started, and the Windows client was installed and connected to the server for monitoring. The process involved various steps including preparing the server, creating configuration files, and installing client software. Velociraptor provides a robust platform for digital forensics and incident response, enabling effective endpoint monitoring and analysis.

Investigating a PUP with Velociraptor

Summary

This report documents the process of infecting a Windows machine with a simple malware sample and investigating the infection using Velociraptor on a Debian Linux server. It includes steps for disabling Windows Defender, installing the malware sample, identifying attack techniques using ATT&CK framework, investigating the incident with Velociraptor, and performing remediation steps.

Introduction:

The purpose of this project is to simulate a malware infection on a Windows machine and analyze the incident from a Linux Velociraptor server. This report outlines the steps taken to infect the Windows machine with a Potentially Unwanted Program (PUP) and investigate the infection using Velociraptor. It also covers the identification of ATT&CK techniques associated with the malware and the execution of remediation steps.

Case Information:

- Server OS: Debian Linux
- Client OS: Windows

Tools Used:

- Web browser
- Velociraptor
- Notepad
- Command Prompt

Procedure:

1. Infecting the Windows Machine:

- **Disabling Windows Defender:**
    - Disabled Windows Defender settings to prevent interference during malware execution.
- **Installing the PUP:**
    - Downloaded and extracted the PUP sample on the Windows machine.

    - Executed the PUP executable as administrator and moved necessary files to system directories.
- **Simulating Malware Execution:**
    - Restarted the Windows machine to trigger the PUP execution.

2. Investigating the Incident with Velociraptor:

- **Connecting to the Client:**
    - Accessed Velociraptor GUI and selected the client's Client ID to view collected data.
- **Using Autoruns:**
    - Utilized the Autoruns collector to identify the Run key used to launch the PUP.
    - Downloaded and analyzed the CSV report to locate the Run key.
- **MD5 of EXE:**
    - Used the Pslist collector to find the MD5 hash of the EXE used to launch the PUP process.
- **Yara Scan:**
    - Launched the Yara collector to search for other EXE files containing specific strings indicative of malware involvement.

3. Remediation:

- **Opening the Shell from Velociraptor:**
    - Accessed the client's Shell from Velociraptor GUI to perform remediation steps.
- **Remediation Commands:**
    - Executed commands to terminate the PUP process, delete related registry keys, and remove the PUP file.

Artifacts and Analysis:

Task	Artifact	Relevance	Tools Used	Description	Evidence Found

Windows Malware Infection	PUP executable	Initiating malware execution	Web browser, Windows	Installed and executed PUP on Windows machine	"YOUR MACHINE IS PWNED"
	Autoruns CSV report	Identifying Run key	Velociraptor, Notepad	Analyzed Autoruns report to locate Run key used for PUP	"PUP4"
	MD5 hash of PUP EXE	Identifying malware executable	Velociraptor	Retrieved MD5 hash of PUP executable used for malware execution	MD5 hash found Flag = "a53"
	Yara Scan Results	Searching for other malware files	Velociraptor	Conducted Yara scan to identify additional malware files on Windows machine	Malware files found Flag = "PWNED"
Remediation	Command Execution Results	Verifying remediation	Velociraptor, Command Prompt	Executed remediation commands and verified their effects on the Windows machine	Remediation verified Flag = "completed successfully"

Conclusion:

The infection of a Windows machine with a simple malware sample and subsequent investigation using Velociraptor on a Debian Linux server was successfully completed. The malware infection was identified and analyzed using various Velociraptor collectors, and remediation steps were executed to remove the malware from the system. Velociraptor proved to be an effective tool for incident response and forensic analysis, providing insights into the attack techniques and enabling prompt remediation actions.

Investigating a Bot with Velociraptor

Summary

This report documents the process of infecting a Windows machine with a simulated Potentially Unwanted Program (PUP) and investigating the infection using Velociraptor on a Debian Linux server. It includes steps for disabling Windows Defender, installing the malware sample, identifying ATT&CK techniques associated with the infection, investigating the incident using Velociraptor, and executing remediation steps.

Introduction:

The purpose of this project is to simulate a malware infection on a Windows machine and analyze the incident from a Linux Velociraptor server. This report outlines the steps taken to infect the Windows machine with a PUP sample, identify ATT&CK techniques used by the malware, investigate the incident using Velociraptor collectors, and perform remediation actions to mitigate the infection.

Case Information:

- Server OS: Debian Linux
- Client OS: Windows

Tools Used:

- Web browser
- Velociraptor
- Wireshark
- Command Prompt

Procedure:

1. Infecting the Windows Machine:

- **Disabling Windows Defender:**
    - Disabled Windows Defender settings to prevent interference during malware execution.
- **Installing the Malware:**
    - Downloaded and extracted the security.zip file containing the malware sample on the Windows machine.

- Executed Bginfo64.exe as administrator to simulate malware execution.

2. Investigating the Incident with Velociraptor:

- **Connecting to the Client:**
  - Accessed Velociraptor GUI and selected the client's Client ID to initiate investigation.
- **Capturing Network Traffic:**
  - Installed Wireshark on the Linux machine to capture network traffic remotely from the Windows client.
  - Launched Windows.Network.PacketCapture collector in Velociraptor to capture network packets.
- **Analyzing Network Traffic:**
  - Downloaded and opened the captured .pcapng file in Wireshark to analyze network traffic.
  - Identified HTTP requests to a subdomain of "samsclass.info" and extracted User-Agent strings as evidence.

3. Further Investigation and Remediation:

- **Analyzing DNS Cache:**
  - Launched Windows.System.DNSCache collector in Velociraptor to analyze DNS cache.
  - Identified the C & C domain name from the DNS cache to uncover additional evidence.
- **Identifying Beaconing EXE:**
  - Utilized Windows.Search.Yara collector with a YARA rule to find files referencing the C & C domain name.
  - Identified the file responsible for beaconing activities and extracted relevant information.
- **Identifying Persistence Mechanism:**
  - Installed Sysmon on the Windows client using Windows.Sysinternals.SysmonInstall collector.
  - Launched Windows.EventLogs.EvtxHunter collector to collect sysmon event logs.

- Analyzed event logs to identify Scheduled Task persistence mechanism and extracted relevant information.

- **Creating and Running Hunts:**
  - Created a new hunt in Velociraptor to detect the securitytest beaconer.
  - Configured parameters for Sysmon installation, event log analysis, DNS cache analysis, and task scheduler analysis.
  - Executed the hunt to identify and analyze suspicious activities on the Windows client.

4. Remediation:

- **Executing Remediation Commands:**
  - Executed remediation commands in an Administrator Command Prompt on the Windows machine to delete scheduled tasks and remove malware-related directories.

Artifacts and Analysis:

Task	Artifact	Relevance	Tools Used	Description	Evidence Found
Windows Malware Infection	Malware executable	Initiating malware execution	Web browser, Windows	Installed and executed PUP on Windows machine	Malware executed
	Network traffic capture	Analyzing malware communication	Velociraptor, Wireshark	Captured and analyzed network traffic to identify malware communication patterns	Beaconing activity Flag = “Mosilla”
	DNS cache analysis	Uncovering C & C domain	Velociraptor	Analyzed DNS cache to identify C & C domain used by malware	C & C domain found Flag = “TTL”
	Beaconing EXE	Identifying beaconing activities	Velociraptor	Identified malware file responsible for beaconing activities	Securitytest.exe Flag = “Upload”

	Sysmon event logs	Identifying persistence mechanism	Velociraptor, Command Prompt	Analyzed Sysmon event logs to identify scheduled task persistence mechanism	Flag = “DcamLaunch”
Hunts and Analysis	Hunt results	Detecting suspicious activities	Velociraptor	Created and executed hunts to detect and analyze suspicious activities on the Windows client	Suspicious activity Flag = “OSPath”
Remediation	Command Execution Results	Verifying remediation	Command Prompt	Executed remediation commands to delete scheduled tasks and remove malware-related directories	Remediation verified

Conclusion:

The investigation into the malware infection on the Windows machine using Velociraptor on a Debian Linux server provided valuable insights into the attack techniques and behavior of the malware. Various ATT&CK techniques were identified, including DLL Search Order Hijacking, Scheduled Task creation, and Tool Impairment. Velociraptor proved to be an effective tool for incident response and forensic analysis, enabling the identification of beaconing activities, persistence mechanisms, and suspicious behaviors. Remediation actions were successfully executed to mitigate the infection and remove the malware from the system.

Investigating a Two-Stage RAT with Velociraptor

Summary

This report outlines the procedures followed to investigate a malware infection on a Windows machine using Velociraptor installed on a Debian 10 Linux server. The investigation involves installing Sysmon, 7-zip, infecting the Windows machine with malware, and conducting various incident response tasks to analyze the infection.

Introduction:

Velociraptor is utilized as the digital forensics platform for this investigation. The purpose is to analyze the malware infection on the Windows machine from a Linux Velociraptor server.

Case Information:

- Linux Server: Debian 11
- Windows Client: Windows 11

Tools Used:

- Velociraptor
- Sysmon
- 7-Zip

Procedure:

1. Installing Sysmon on the Windows Machine:

- Connected to the Velociraptor client from the Linux server.
- Launched Velociraptor GUI and navigated to the client's Client ID.
- Executed the collector "Windows.Sysinternals.SysmonInstall" to install Sysmon on the Windows machine.

2. Installing 7-Zip on the Windows Machine:

- Downloaded and installed the 64-bit version of 7-Zip from the official website.

3. Infecting the Windows Machine:

- Downloaded the malware file "pup5.zip" using Microsoft Edge.
- Extracted the file using the password "malware".

- Ran "pup5.bat" as Administrator, bypassing security warnings.

4. Investigating the Incident with Velociraptor:

Auditing Network Connections

- Launched the collector "Windows.Network.Netstat" to audit network connections.
- Identified the port the "shellbind" process is listening on.

Auditing Autoruns

- Launched the collector "Windows.Sysinternals.Autoruns" to audit autoruns.
- Searched for "shellbind.exe" in the results and noted its path.

Creation Time

- Launched the collector "Windows.System.PowerShell" to examine creation time.
- Conducted a directory listing of the folder containing "shellbind.exe" and noted the creation timestamp.

Prefetch

- Launched the collector "Windows.Forensics.Prefetch" to examine prefetch data.
- Noted the LastRunTimes associated with "shellbind.exe".

Sysmon Logs

- Launched the collector "Windows.EventLogs.Evtx" to collect Sysmon event logs.
- Restricted the time to the time of the event related to malware execution.
- Examined the events to identify the event associated with the execution of "pup5.bat" and the subsequent extraction and decryption of shellcode.

Artifacts and Analysis:

Task	Artifact	Relevance	Tools Used	Description	Evidence Found
Network Connections	Network connections	Indication of communication	Velociraptor	Identified port of "shellbind" process	Port number = 4444
Autoruns	Autoruns	Indication of persistence	Velociraptor	Noted path of "shellbind.exe"	File path = C:\\shellbind.ex

Creation Time	Creation time	Indication of malware installation	Velociraptor	Noted creation timestamp	5/11/24 6:13 PM
Prefetch	Prefetch data	Indication of program execution	Velociraptor	Noted LastRunTimes	5/11/24 6:13 PM Flag = “Hash”
Sysmon logs	Sysmon event logs	Indication of system activity	Velociraptor	Identified relevant events	Company = “Igor Pavlov”

Conclusion:

The investigation of the malware infection using Velociraptor on the Linux server proved to be comprehensive. Various incident response tasks were conducted to analyze the infection and identify relevant artifacts. The examination provided valuable insights into the nature of the malware and its impact on the Windows machine.