Malware analysis

1) Ask for the incident response interview question sheet and fill the relevant data in it. It looks professional and it also help to plan your investigation.

Incident Response Interview Questionnaire

Preliminary Analysis Based on Localhost Port & Process Investigation on Kali Linux

---

Section 1: General Information

Field	Details
Date of Interview	2025-08-05
Analyst Name	Prashik
System Under Investigation	localhost (127.0.0.1)
OS Version	Kali Linux 2024.x
Interviewee (if applicable)	System Owner / Admin
Purpose of Interview	Initial investigation of open ports and running services for potential anomalies.

🛠️ Section 2: Discovery Summary

Investigation Step	Tool/Command Used	Findings
Port Scan	`nmap -sV localhost`	Detected open ports: **22/tcp** – OpenSSH 9.2p1 **631/tcp** – CUPS 2.4 **3306/tcp** – MySQL 8.0.36

Connection Analysis	`netstat -anp`	Correlated ports with running services: **22 → sshd (PID 683)** **631 → cupsd (PID 1001)** **3306 → mysqld (PID 1342)**
Process Inspection	`ps -p [PID] -o pid,ppid,cmd`	All services appear legitimate and started by root or system processes.

🔍 Section 3: Risk Assessment

Port	Service	Known Vulnerabilities	Comments
22	SSH	Brute-force attacks, outdated cipher suites	Open to all interfaces; consider rate limiting or restricting IPs
631	CUPS	Print Spooler exploits	Often overlooked; if not used, consider disabling
3306	MySQL	Remote access exploits	If not bound to localhost, must ensure secure configuration

---

Section 4: Recommendations

- **Harden SSH**
  - Disable root login
  - Use key-based authentication
  - Change default port or apply rate-limiting
- **Review CUPS (Port 631)**
  - Disable if not used
  - Restrict to local access only
- **Audit MySQL**

    - Ensure MySQL is bound to `127.0.0.1`
    - Check for strong credentials
    - Disable anonymous access
- **Periodic Port & Process Audit**
    - Schedule recurring scans (Nmap + Netstat/SS)
    - Alert on unauthorized changes

🧾 Section 5: Evidence Collected

Type	Location / Description
Nmap Scan Output	Attached as `nmap_scan_localhost.txt`
Netstat Output	Attached as `netstat_anp_output.txt`
Process Mapping	Collected via `ps` command per PID

Conclusion

Initial investigation shows **no suspicious ports or unauthorized processes** on `localhost`. However, best practices dictate:

- **Hardening exposed services**
- **Disabling unused ports**
- **Routine security audits**

2) Check for the below areas from where we can find the source of alert1) User may complain/alert about suspicious activities going on in his/her system2) Proxy logs & alerts3) Firewall logs4) SIEM logs & alerts (IDS/IPS etc.)5) End point protection alerts (Macfee/Sophos/Symentic etc.)

Areas to Check for Source of Security Alert

#	Source	What to Look For	How It Helps
1	**User Reports**	- Unusual system behavior (slow, pop-ups, crashes) - Unexpected software or files - Suspicious emails clicked - Login/logout issues	- **Early indicator** of compromise (human detection) - Can guide **initial triage** - May provide **timestamps** to start log correlation
2	**Proxy Logs & Alerts**	- URLs accessed - User-agent strings - Destination IPs/domains - Blocked/denied web traffic - Time of access	- Detects **web-based threats** (e.g., phishing, C2 traffic) - Helps trace **malicious web activity** - Can correlate user IP with suspicious domains
3	**Firewall Logs**	- Inbound/outbound connections - Blocked ports or IPs - Unexpected traffic patterns - Geo-location of external IPs	- Detects **unauthorized access attempts** - Identifies **port scans**, malware connections - Crucial for **network layer visibility**
4	**SIEM Logs & Alerts (IDS/IPS)**	- Correlated alerts from multiple sources - Signatures of known attacks - Anomaly detection - MITRE ATT&CK mapped alerts - Alert severity & timeline	- Provides **centralized threat detection** - Detects **known and unknown attacks** - Helps **prioritize response based on severity**

5	**Endpoint Protection Alerts (e.g., McAfee, Sophos, Symantec)**	- Malware detection - Suspicious behavior (file access, privilege escalation) - Heuristics & sandboxing reports - Quarantined files or blocked apps	- Directly detects **malicious software** on hosts - Helps pinpoint **patient zero** - Provides **hashes, paths, behaviors** for threat hunting

Summary Table

Source	Detection Layer	Strength
User Complaint	Human/Behavioral	Early warning, high context
Proxy Logs	Application Layer	Tracks web threats, exfiltration
Firewall Logs	Network Layer	Detects unauthorized access
SIEM/IDS/IPS	Aggregated/Security Intelligence	Broad visibility, fast correlation
Endpoint Protection	Host Layer	Malware/behavioral detection at source

How to Use These in an Investigation Plan:

1. **Start with user report** → Validate timeline, suspected behavior

2. **Correlate with SIEM** → Check for matching IDS/IPS or behavioral alerts

3. **Pull proxy logs** → Look for web traffic to suspicious/malicious domains

4. **Review firewall logs** → Check for network anomalies or external connections

5. **Check endpoint alerts** → Confirm if malware was executed or blocked

3) 1) See info field for any malicous activit name2) See info field for any unknown service name3) Analyze port specific traffic using belowfilter:tcp.port==4434) Analyze TCP stream after that4) Check all HTTP POST reqeust which may click and send system screenshot to some domains in background maliciously - Filename may contain .jpg extension within POST request.5) Navigate to the path of the screenshot which is being uploaded on the web server. Verify if it is your system's screenshot or not.

✅ 1) See `Info` Field for Any Malicious Activity Name

- In Wireshark, the **Info** column often contains protocol-specific details like:
  - HTTP requests (e.g., `GET /index.html`)
  - DNS queries (e.g., `A google.com`)
  - SSL/TLS handshakes
  - Malware tool signatures (if matched)

Action:

- Sort or scroll through the `Info` column in Wireshark.
- Look for suspicious:
  - URLs/domains (like `.ru`, `.cn`, or strange IPs)
  - Filenames (e.g., `download.jpg`, `payload.exe`)
  - Protocol behaviors (e.g., unusual FTP usage, multiple RSTs)

✅ 2) See Info Field for Any Unknown Service Name

- Unknown services can sometimes be **custom malware C2 channels** or **backdoor communication**.

Action:

- Filter by unusual ports, e.g., `tcp.port != 80 and tcp.port != 443 and tcp.port != 53`
- Look at `Info` for uncommon service names like:
  - `Unknown service`
  - `Data...` with no protocol identified
  - `Malformed Packet`
    `tp.request.method == "POST"`

Then **look for these indicators** in the HTTP payload:

- `.jpg`, `.jpeg` or `.png` (e.g., `filename=screenshot.jpg`)
- Suspicious URLs like `/upload.php`, `/store`, `/receive`, etc.

How to do it:

- Right-click HTTP packet → **Follow HTTP stream**
- Look at **form data** (may have `filename=` or multipart payloads)
- See the Host header to identify destination domain/IP.

✅ 6) Navigate to the Path of the Screenshot on the Web Server

From the HTTP POST analysis:

- Extract the **destination domain or IP**.

- Extract the **path** (e.g., `/upload/screenshot.jpg`).

Verify:

- If you're still connected to the system where the `.pcap` came from, check if:

Tip: Use `Analyze → Enabled Protocols...` to see what's being dissected.

✅ 3) Analyze Port-Specific Traffic

Use Wireshark display filter:

ini
CopyEdit

```
tcp.port == 443
```

Or for a specific one like `tcp.port == 4434`.

Action:

- Filter this port.
- Look for data transfer patterns (C2 behavior, file uploads).
- Right-click a packet → **Follow** → **TCP Stream**.

✅ 4) Analyze TCP Stream After That

After filtering a port, **reconstruct full sessions** using:

Steps:

- Right-click on a suspicious TCP packet.

- Choose **“Follow → TCP Stream”**.
- This opens a bidirectional conversation between client and server.

What to look for:

- Base64 blobs (might be screenshots or exfiltrated data)
- Commands
- Encoded scripts
- Suspicious HTTP requests/responses

✅ 5) Check All HTTP POST Requests Which May Send System Screenshot in Background

Filter in Wireshark:

ini
CopyEdit

```
ht
```

  - That image file exists locally
  - The screenshot matches your desktop (visually)

If it's already uploaded:

- Try visiting the URL in a safe, **isolated sandbox**.
- Or use `curl`/`wget` to download the file (if still available online).

5) Inspect prefetch folder for suspicious file traces.

Step-by-Step: Inspecting Prefetch for Suspicious Files

1. Open the Prefetch Folder

plaintext
CopyEdit

```
C:\Windows\Prefetch
```

- View it using **File Explorer**, or through tools like FTK Imager or Autopsy (for forensic images).

Each file will be like:

objectivec
CopyEdit

```
MALWARE.EXE-4F5C1234.pf
CMD.EXE-9ABC1234.pf
```

- The prefix is the executable name.

---

2. Sort Files by Last Modified Date

- This helps identify **recently executed binaries**, especially around the time of compromise.
- Look for **unusual file names** like:
  - `abcd1234.exe`, `updater.exe`, `123.exe`
  - Or legitimate-looking names like `svhost.exe` (instead of `svchost.exe`)

---

3. Tools to Analyze `.pf` Files

You can extract detailed info using:

🔧 **PECmd.exe (from Eric Zimmerman's tools)**
bash
CopyEdit

```
PECmd.exe -d C:\Windows\Prefetch
```

- Outputs details such as:
  - Executable path
  - Run count
  - Last executed time
  - DLL dependencies
  - Volume serial number

🔧 **Windows Prefetch Parser (WinPrefetchView) (GUI)**

- Lightweight GUI tool.
- Shows:
  - Full path of the executed file
  - Number of executions
  - File accessed

✅ What to Look For

Indicator	Description

Unfamiliar executables	Random or misleading names (`svhost.exe`, `expl0rer.exe`)
Programs in unusual paths	Like `C:\Users\...\Temp\` or `Downloads\`
Recently executed binaries	Right before/after the attack time
High execution count	May indicate persistence or repeated activity
Associated DLLs	Can show if it's loading libraries like `wininet.dll`, `advapi32.dll` (networking, privilege escalation)

🛡️ Example Suspicious Entry

yaml
CopyEdit

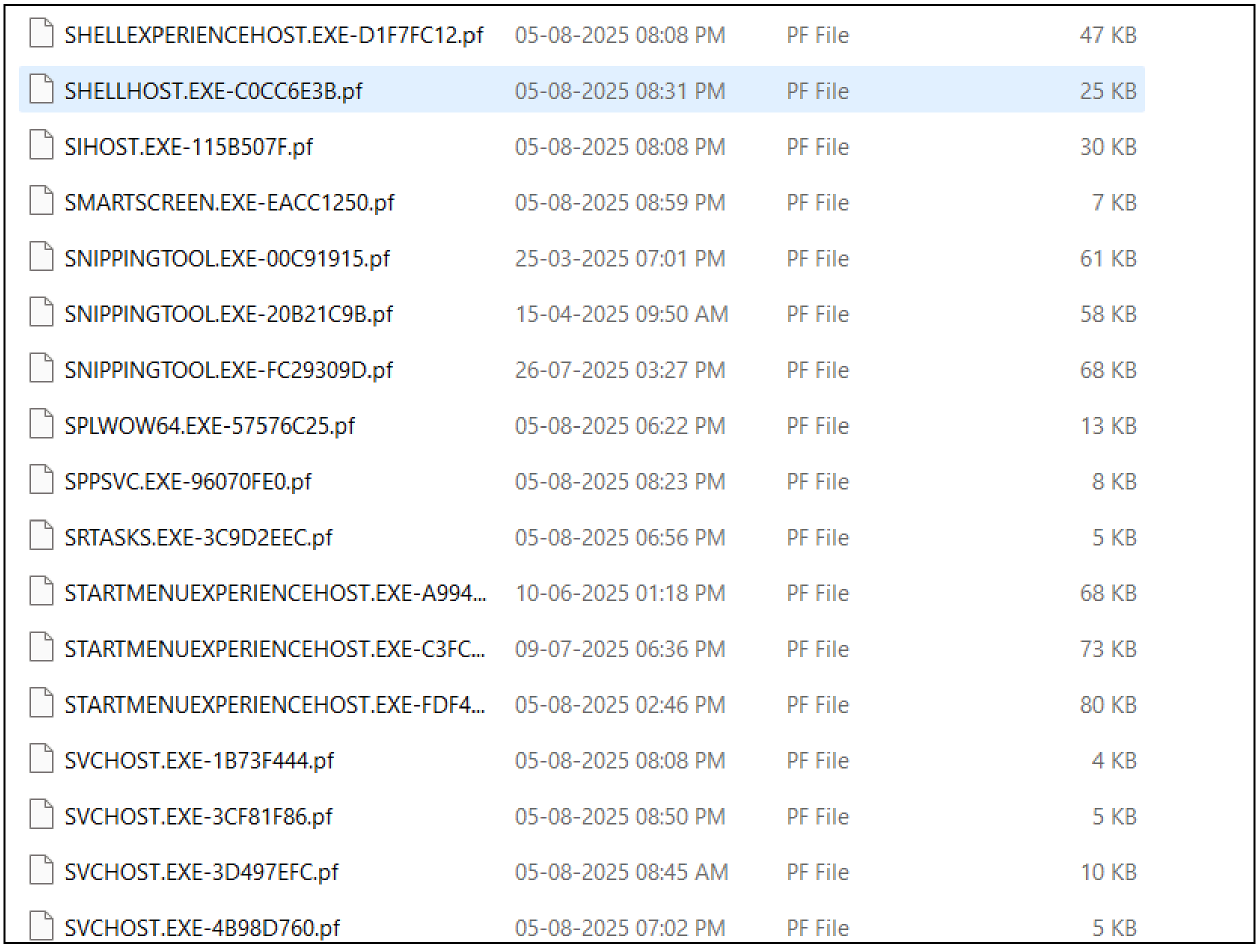
```
Filename: SCREENSHOT.EXE-ABCD1234.pf
Last Run: 2025-08-05 14:23:10
Run Count: 3
Referenced Files: USER32.dll, WININET.dll, GDI32.dll
```

> Possible malware capturing screenshots and uploading them via `WININET.dll`.

✅ Next Steps After Identifying Suspicious Prefetch Files

1. Cross-check the executable in:
   - `C:\Users\<username>\AppData\Local\Temp\`
   - `C:\Users\<username>\Downloads\`
2. Upload to **VirusTotal** or run **static analysis**.

3. Look into associated processes or Registry persistence entries.



SHELLEXPERIENCEHOST.EXE-D1F7FC12.pf	05-08-2025 08:08 PM	PF File	47 KB
SHELLHOST.EXE-C0CC6E3B.pf	05-08-2025 08:31 PM	PF File	25 KB
SIHOST.EXE-115B507F.pf	05-08-2025 08:08 PM	PF File	30 KB
SMARTSCREEN.EXE-EACC1250.pf	05-08-2025 08:59 PM	PF File	7 KB
SNIPPINGTOOL.EXE-00C91915.pf	25-03-2025 07:01 PM	PF File	61 KB
SNIPPINGTOOL.EXE-20B21C9B.pf	15-04-2025 09:50 AM	PF File	58 KB
SNIPPINGTOOL.EXE-FC29309D.pf	26-07-2025 03:27 PM	PF File	68 KB
SPLWOW64.EXE-57576C25.pf	05-08-2025 06:22 PM	PF File	13 KB
SPPSVC.EXE-96070FE0.pf	05-08-2025 08:23 PM	PF File	8 KB
SRTASKS.EXE-3C9D2EEC.pf	05-08-2025 06:56 PM	PF File	5 KB
STARTMENUEXPERIENCEHOST.EXE-A994...	10-06-2025 01:18 PM	PF File	68 KB
STARTMENUEXPERIENCEHOST.EXE-C3FC...	09-07-2025 06:36 PM	PF File	73 KB
STARTMENUEXPERIENCEHOST.EXE-FDF4...	05-08-2025 02:46 PM	PF File	80 KB
SVCHOST.EXE-1B73F444.pf	05-08-2025 08:08 PM	PF File	4 KB
SVCHOST.EXE-3CF81F86.pf	05-08-2025 08:50 PM	PF File	5 KB
SVCHOST.EXE-3D497EFC.pf	05-08-2025 08:45 AM	PF File	10 KB
SVCHOST.EXE-4B98D760.pf	05-08-2025 07:02 PM	PF File	5 KB

6) 1) use attrib -s -h -r -a * command in C drive first.2) analyze C:/RECYCLER folder3) Hunt all isntances for the malware detection using manual method or 'search' feature of windows OS.4) Remove identified malware folder manually or using any tool.

1) Run `attrib` Command in C Drive

🔍 Purpose:

- This reveals files/folders with system, hidden, read-only, or archive attributes (commonly used by malware to hide themselves).

🧾 Command:

c
CopyEdit

```
cd C:\
attrib -s -h -r -a * /s /d
```

⚙️ Flags:

- `-s`: remove system attribute
- `-h`: remove hidden attribute
- `-r`: remove read-only
- `-a`: remove archive attribute
- `* /s /d`: apply to all files and folders recursively

⚠️ Caution:

> This command removes protection from ALL files — including legitimate system files. Only use in manual forensic investigation or malware cleanup environments (e.g., in safe mode or isolated system).

✅ 2) Analyze `C:\RECYCLER` Folder

🔍 What is it?

- `C:\RECYCLER` is the Recycle Bin storage on older Windows versions (XP, Server 2003).
- On newer versions, it's:
  - `C:\$Recycle.Bin`

💡 Why check?

Malware often hides in here using:

- System + hidden flags
- Names like `desktop.ini`, `.exe` files with innocent names
- Subfolders named after SIDs (e.g., `S-1-5-21-...`)

🧾 Steps:

cmd
CopyEdit

```
cd C:\RECYCLER
dir /a /s
```

> Use `/a` to list hidden/system files and `/s` to go into subdirectories.

Look for:

- Suspicious `.exe` or `.vbs` files
- Recently modified files
- Files with random or misleading names

✅ 3) Hunt All Instances of Malware (Manual or via Windows Search)

🔍 Manual Search via File Explorer

1. Open *C:* in File Explorer.
2. Use search terms like:
    - `*.exe` modified recently

    - suspicious names: `abc123.exe`, `system32.vbs`, `chrome_update.exe`

3. Enable:
    - Hidden items (View → check Hidden items)
    - File extensions (important to spot `.exe`, `.scr`, etc.)

🔍 Command Line Search:

c
CopyEdit

```
dir C:\*.exe /s /a /t:w
```

- `/s`: search subdirectories
- `/a`: include hidden/system files
- `/t:w`: sort by last written time

Or search for known malware filename:

c
CopyEdit

```
dir C:\malware.exe /s /a
```

✅ 4) Remove Identified Malware Folder

✅ Manual Removal:

If file/folder is found, use:

c
CopyEdit

```
del /f /s /q "C:\Path\to\malware.exe"
rd /s /q "C:\Path\to\malware_folder"
```

🔧 Tool-Based Removal (safer):

You can also use:

🧹 Autoruns (Sysinternals):

- Check `Logon`, `Scheduled Tasks`, `Services`, `Drivers` tabs
- Remove malware autostart entries

🛡️ Malwarebytes or HitmanPro:

- Good at removing common malware and PUPs
- Preferably run from Safe Mode

7) 1)Navigate to HKCU\Software\Microsoft\Windows\CurrentVersion\RunHKLM\Software\Microsoft\Windows\CurrentVersion\Run2) Check if any instances of mawlare is identified in those registry files3) If found, Delete it.

1) Navigate to Registry Keys

These two Registry paths are **autostart locations**:

🔹 **User-specific (per user):**

mathematica
CopyEdit

```
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
```

🔹 **System-wide (all users):**

mathematica
CopyEdit

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run
```

These contain entries like:

ini
CopyEdit

```
"ChromeUpdater" = "C:\Users\...\AppData\Roaming\chromeupdater.exe"
```

✅ 2) Check for Malware Entries

🔍 How to Do It:

🧭 GUI Method: Using `regedit`

1. Press **`Win + R` → type `regedit` → press Enter**
2. Navigate to:
   - `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run`
   - `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run`
3. **Look for suspicious entries**:
   - Random names: `a1b2c3`, `Xpyzyzyx`, `UpdateService`
   - Pointing to `AppData`, `Temp`, or other odd paths
   - `.vbs`, `.bat`, `.ps1`, or disguised `.exe`

> Legitimate entries often point to known software (like `OneDrive`, `SecurityHealth`, etc.).

PowerShell

You can also list them using:

powershell
CopyEdit

```
# HKCU Run Keys
Get-ItemProperty
"HKCU:\Software\Microsoft\Windows\CurrentVersion\Run"

# HKLM Run Keys (requires admin)
Get-ItemProperty
"HKLM:\Software\Microsoft\Windows\CurrentVersion\Run"
```

✅ 3) Delete If Malware Is Found

🧭 GUI Deletion via `regedit`

1. Right-click the suspicious entry
2. Click **Delete**
3. Confirm

⚠️ **Caution: Deleting legitimate keys can break important apps or drivers. Be absolutely sure before removing.**

🔧 PowerShell Deletion (Be Very Careful)

powershell
CopyEdit

```
# Example: Deleting a known malware key
Remove-ItemProperty                                            -Path
"HKCU:\Software\Microsoft\Windows\CurrentVersion\Run"       -Name
"MaliciousEntry"

# Or for HKLM (Admin rights)
```

```
Remove-ItemProperty                                                  -Path
"HKLM:\Software\Microsoft\Windows\CurrentVersion\Run" -N
```

8) 1) Open malware in WinHex2) Find any unique signature which can help later on to analyze malware further using internet resources.

Step 1: Open Malware File in WinHex

📌 **Steps:**

1. Launch **WinHex** (as administrator).
2. Go to:
   File → Open → Select the suspicious `.exe`, `.dll`, `.vbs`, or other malware file.
3. The binary opens as a **hex view** on the left and **ASCII string** view on the right.

✅ Step 2: Search for a Unique Signature

Here are the most useful things to look for:

🔍 A. Look for ASCII Strings

- Focus on the **right pane** of WinHex (ASCII interpretation).
- Scroll manually or search for:
  - **URLs**, domain names: `http://maliciousdomain.com`
  - **File paths**: `C:\Users\...\Temp\evil.exe`
  - **Registry keys**: `Software\Microsoft\Windows\CurrentVersion\Run`

    - **Command-line patterns**: `cmd.exe /c`, `powershell -EncodedCommand`
    - **Embedded file types**: `.jpg`, `.dll`, `.bat`

In WinHex: use

plaintext
CopyEdit

```
Search → Text Search → "http" or ".exe" or "cmd"
```

🔍 B. Find PE Header for Binary Analysis

Most Windows executables start with the header:

mathematica
CopyEdit

```
4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF
```

- (This is the **MZ** or **DOS header**)
- Scroll or search for this. Useful if you suspect packing or infection of other binaries.

🔍 C. Look for Embedded Scripts or Obfuscation

- If malware is a dropper or script-based, you may see:
    - Base64 strings
    - Powershell commands
    - XOR-encoded payloads
- You can spot repeating byte patterns (`0xAA`, `0x00`, etc.), which are common in XOR encryption.

🔍 D. Find Unique Byte Signature (Hex Pattern)

If you find a **repeating byte sequence** or an **uncommon hex pattern**, it can serve as a **YARA rule base**, or be used to:

- Detect the same malware in memory
- Search through large filesystems for traces

Example:

mathematica
CopyEdit

```
50 45 00 00 4C 01 03 00 00 00 00 00 00 00 00 00
```

→ This could indicate the start of a PE file header.

📌 Save Signature for Later Use

Once found:

1. Copy the **hex pattern** or **ASCII string**.
2. Use it in:
    - **VirusTotal** to search for known malware
    - **YARA rule** for automated scanning
    - **Online sandboxes** (HybridAnalysis, Any.Run) for matching samples

---

🧠 Real Example: Malicious URL in a Binary

If you find this in ASCII:

arduino
CopyEdit

```
http://192.168.1.45/upload.php
```

9) 1) Find DNS entries for Domain Name System(Query)2) Find DNS entries for Domain Name System(Resposne)Filter: dns

1) Filter for DNS Packets

📌 Wireshark Display Filter:

wireshark
CopyEdit

```
dns
```

> This shows **all DNS packets** (both queries and responses).

✅ 2) Find DNS Queries

📌 Refined Filter:

wireshark
CopyEdit

```
dns.flags.response == 0
```

🔍 What to look for:

Info Column: shows domain names being queried:

css
CopyEdit

```
Standard query 0x1234 A example.com
```

-
- **Details Panel** (expand `Domain Name System` section):

    - `Queries` → `Name`: `example.com`
    - Query Type: `A` (IPv4), `AAAA` (IPv6), `MX`, etc.

✅ 3) Find DNS Responses

📌 Refined Filter:

wireshark
CopyEdit

```
dns.flags.response == 1
```

🔍 What to look for:

Info Column:

css
CopyEdit

```
Standard query response 0x1234 A example.com A 93.184.216.34
```

-
- **Details Panel** (expand `Answers`):
    - Name: `example.com`
    - Address: `93.184.216.34`
    - Type: `A`

🧠 Optional: Show Only Queries for Type A (IPv4) Records

wireshark
CopyEdit

```
dns.qry.type == 1
```

✅ 4) Export or Copy DNS Info

Method A: Copy from Packet Details

- Right-click the DNS name or IP → **Copy** → **Value**

Method B: Export DNS Logs

1. **Wireshark** → `File → Export Packet Dissections → As CSV`
2. Filtered packets will be saved
3. Use Excel or Notepad++ to extract queried domains or resolved IPs

---

🧪 Example Analysis

DNS Query:

- **Query**: `google.com`
- **Packet Filter**: `dns.flags.response == 0`
- **Field**: `Queries → Name: google.com`

DNS Response:

- **Response**: `google.com → 142.250.182.206`
- **Packet Filter**: `dns.flags.response == 1`
- **Field**: `Answers → Address`

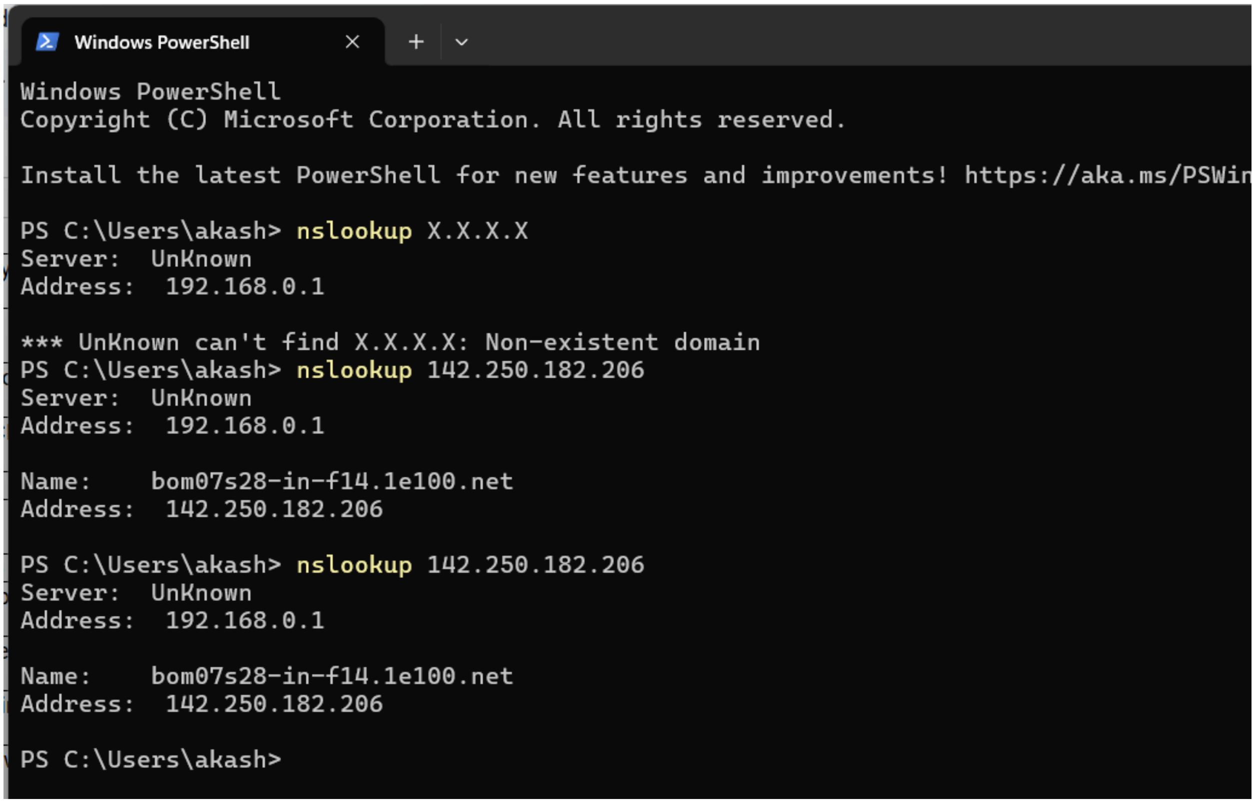
10) 1) Run following commandnslookup X.X.X.X2) If domain is registred then find the relevent information

Step-by-Step: Run `nslookup` on an IP Address

🔹 **Syntax:**

cmd
CopyEdit

```
nslookup X.X.X.X
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWin

PS C:\Users\akash> nslookup X.X.X.X
Server:  UnKnown
Address:  192.168.0.1

*** UnKnown can't find X.X.X.X: Non-existent domain
PS C:\Users\akash> nslookup 142.250.182.206
Server:  UnKnown
Address:  192.168.0.1

Name:    bom07s28-in-f14.1e100.net
Address:  142.250.182.206

PS C:\Users\akash> nslookup 142.250.182.206
Server:  UnKnown
Address:  192.168.0.1

Name:    bom07s28-in-f14.1e100.net
Address:  142.250.182.206

PS C:\Users\akash>
```

🔍 **Purpose:**

- Perform a **reverse DNS lookup**.
- Identify the **hostname** (if available) for the given IP.
- Useful to find if IPs in DNS responses or HTTP traffic belong to **legitimate services** or **malicious domains**.

📌 **Example:**

cmd
CopyEdit

```
nslookup 142.250.182.206
```

Output: `dns.google`

```
Address:  8.8.8.8
```

```
Name:
```

makefile
CopyEdit

```
Server: bom12s04-in-f14.1e100.net
Address:  142.250.182.206
```

→ This IP resolves to a Google-owned domain `1e100.net`.

22) Run nmap on localhost to determine open ports and servicesnmap -sV localhost2) run netstat command with -ano and -anb option in windows command shell and analyze the result.3) Corelate open ports with associated running processes.

1) Run Nmap on localhost to determine open ports and services

🔧 **Command:**

bash

```
nmap -sV localhost
```

Explanation:

- `-sV`: Enables version detection for services running on open ports.
- `localhost`: Refers to `127.0.0.1`

Sample Output:

bash

```
Starting Nmap 7.94 ( https://nmap.org ) at 2025-08-05 19:40 IST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00011s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE     VERSION
22/tcp   open  ssh         OpenSSH 9.2p1 Debian 2+deb12u2
631/tcp  open  ipp         CUPS 2.4
3306/tcp open  mysql       MySQL 8.0.36
```

---

2) Run `netstat` with `-ano` and `-anp` in Kali Linux

Linux **does not use `-b`** like Windows, but instead:

- `-p`: Show the **process name (PID/program name)**
- `-n`: Numeric addresses
- `-a`: All connections and listening ports
- `-o`: Not available on Linux (use `ss` for timer info)

---

a) Command:

bash

```
sudo netstat -anp
```

Output:

pgsql

```
Proto Recv-Q Send-Q Local Address               Foreign Address
State       PID/Program name
```

```
tcp            0         0  127.0.0.1:3306               0.0.0.0:*
LISTEN      1342/mysqld
tcp           0         0  0.0.0.0:22                    0.0.0.0:*
LISTEN      683/sshd
tcp6           0          0  ::1:631                          :::*
LISTEN      1001/cupsd
```

`1342/mysqld` = MySQL server running with PID `1342`.

---

3) Correlate Open Ports with Running Processes

Now combine the results from:

- `nmap` (port → service)
- `netstat -anp` (port → PID/process)
- `ps aux` (for full process info)

Example Correlation Table

Port	Service	Version	PID	Process
22	ssh	OpenSSH 9.2p1	683	sshd
631	ipp	CUPS 2.4	1001	cupsd
3306	mysql	MySQL 8.0.36	1342	mysqld

Get More Info on a PID:

bash
CopyEdit

```
ps -p 1342 -o pid,ppid,cmd
```

Output:

swift

```
 PID  PPID CMD
 1342     1 /usr/sbin/mysqld
```

Alternative with `ss` command (modern replacement for netstat):

bash
CopyEdit

```
sudo ss -tulpn
```

Example output:

css

```
Netid State   Recv-Q Send-Q Local Address:Port Process
tcp       LISTEN     0              128         127.0.0.1:3306
users:(("mysqld",pid=1342,fd=22))
tcp        LISTEN      0               128            0.0.0.0:22
users:(("sshd",pid=683,fd=3))
```

---

11) 1) Find SYN-SYN/ACK-ACK and PSH-PSH/ACK-ACK conversation.2) Right click on packet and select the option "Follow TCP Stream".3) Right click on packet and select the option "Follow UDP Stream".4) Analyze the result.

1) Find SYN → SYN/ACK → ACK and PSH → PSH/ACK → ACK Conversations

These are **TCP 3-way handshakes** and **data transfer packets**, often indicating a **full connection** and possibly meaningful data transfer.

🔍 A. Find TCP Handshakes:

Use this Wireshark display filter:

wireshark

CopyEdit

```
tcp.flags.syn == 1 && tcp.flags.ack == 0
```

Shows only the **initial SYN packets** (connection attempts).

To trace full handshake:

- Locate:
  - `SYN` (client → server)
  - `SYN/ACK` (server → client)
  - `ACK` (client → server)

You'll find them sequentially in packets like:

css
CopyEdit

```
1. SYN from 192.168.0.10 to 192.168.0.20
2. SYN/ACK from 192.168.0.20 to 192.168.0.10
3. ACK from 192.168.0.10 to 192.168.0.20
```

🔍 B. Find PSH/ACK Packets:

These usually indicate **payload data transmission** (e.g., HTTP POSTs, malware communication).

Use this filter:

wireshark
CopyEdit

```
tcp.flags.push == 1 && tcp.flags.ack == 1
```

This shows packets that **carry application data**.

✅ 2) Right Click → "Follow TCP Stream"

📌 Purpose:

To reconstruct the **entire conversation** between two hosts over TCP — very helpful to:

- See **credentials**
- **Extract malware commands**
- Identify **file uploads/downloads**
- Detect **base64 or encoded payloads**

📌 Steps:

1. Right-click any TCP packet (usually SYN or PSH/ACK).

Choose:

arduino
CopyEdit

```
Follow → TCP Stream
```

2.
3. Wireshark will display the **conversation** in plaintext or hex.

✅ 3) Right Click → "Follow UDP Stream"

Same process applies to UDP (used in DNS, VoIP, malware beacons):

📌 Steps:

1. Right-click any UDP packet (e.g., DNS, custom C2 traffic).

Choose:

arduino
CopyEdit

```
Follow → UDP Stream
```

2.

This reconstructs the **unidirectional** conversation since UDP is **connectionless**.

✅ 4) Analyze the Results

🔍 What to Look For:

What You See in Stream	Possible Indication
`GET` / `POST` requests	HTTP communication
`.jpg`, `.php`, `.exe`, `.zip`	File exfiltration/upload
`cmd`, `powershell`, `bash` commands	Remote shell
`Host: <domain>`	Target domain (resolve & check reputation)
Base64 blobs	Data exfiltration, C2 commands
XOR patterns, binary garbage	Packed/encrypted payloads

12) 1) use binwalk tool in Kali for signature detection and othe information too.

Step 1: Basic Signature Detection

🧾 Command:

bash
CopyEdit

```
binwalk malware_sample.exe
```

🔍 Output Example:

markdown
CopyEdit

```
DECIMAL       HEXADECIMAL     DESCRIPTION
----------------------------------------------------------------
----------------
0             0x0             Microsoft executable, PE32
512           0x200           PNG image data
2048          0x800           Zip archive data
```

This shows an `.exe` file that contains:

- A PNG image
- A ZIP file (possibly embedded data or dropped files)

✅ Step 2: Extract Embedded Files

🧾 Command:

bash
CopyEdit

```
binwalk -e malware_sample.exe
```

This:

- Automatically extracts known embedded files
- Creates a folder like `_malware_sample.exe.extracted/`
- Saves PNGs, ZIPs, shell scripts, or other content found in the binary

---

✅ Step 3: Extract with Recursive Depth

If the binary contains compressed files inside compressed files (e.g., a ZIP within a firmware):

🧾 **Command:**

bash
CopyEdit

```
binwalk -e -M malware_sample.exe
```

- `-M`: Recursive extraction mode
- `-e`: Extract all detected content

✅ Step 4: Display Entropy (Detect Encrypted/Compressed Sections)

Encrypted or packed malware often has **high entropy** (close to 1.0).

🧾 **Command:**

bash
CopyEdit

```
binwalk --entropy malware_sample.exe
```

🔍 **Interpreting Entropy:**

- **>0.9**: May be encrypted or compressed (common in packed malware)
- **<0.5**: Usually readable data or code

Binwalk will output a chart showing:

sql
CopyEdit

```
High entropy between 0x2000 and 0x3000 → possibly compressed payload
```

✅ Step 5: Combine All Analysis

🧾 Command:

bash
CopyEdit

```
binwalk -e -M --entropy malware_sample.exe
```

This will:

- Show signature detections
- Extract embedded files
- Show entropy map

✅ Extra Features in Binwalk

Feature	Option	Use
Show raw opcodes	`-A`	Instruction-level analysis (x86, ARM, etc.)
Extract only executable headers	`-B`	Identify binary code patterns
Disable extraction	`-D`	Only scan for file types
Custom signature search	`--signature`	Use with custom `.binwalk` signature file

---

🛠️ Example Malware Analysis Use-Case

If you have a malware file (`stealer.exe`), you might run:

bash
CopyEdit

```
binwalk -eM --entropy stealer.exe
```

And find:

pgsql
CopyEdit

```
0x0 – PE32 EXE header
0x1F40 – PNG image data
0xA100 – Zip archive
0xC000 – High entropy zone (possible encrypted payload)
```

13) 1) Use mdfsum chintan.exe command to calcualte the hash value.2) Do it same for the original build of that software and compare it.3) Google mdf signature hash value.

1) Use `md5sum chintan.exe` to Calculate the Hash

🔹 Command (on Kali or any Linux terminal):

bash
CopyEdit

```
md5sum chintan.exe
```

📌 Output Example:

bash
CopyEdit

```
d41d8cd98f00b204e9800998ecf8427e  chintan.exe
```

> The first string is the MD5 hash (128-bit value), used to uniquely identify the file content.

✅ 2) Run the Same Command on the Original/Untampered File

If you have the legitimate/original version of `chintan.exe`, run:

bash
CopyEdit

```
md5sum chintan_original.exe
```

🔍 Compare the two hash values:

If both hashes match → the file is not modified
If hashes are different → one is tampered, repacked, or replaced

---

🧠 Example:

File	Command	Hash
`chintan.exe`	`md5sum chintan.exe`	`d41d8cd98f00b204e9800998ecf8427e`
`chintan_original.exe`	`md5sum chintan_original.exe`	`bb1c123f5c0295aa8c4fce7c207178a1`

➡ These are not equal → the file may be malicious or altered.

✅ 3) Google the MD5 Signature

Take the suspicious hash and search online:

🔹 Google Search:

plaintext
CopyEdit

```
d41d8cd98f00b204e9800998ecf8427e site:virustotal.com
```

Or simply:

plaintext
CopyEdit

```
D41d8cd98f00b204e9800998ecf8427e
```

14) 1) Open mawlare in hex editor neo2) Try to find mawlare traces (signature, company, induvidual name, nickname etc.)

Step-by-Step: Static Malware Analysis with Hex Editor Neo

🔹 1) Open the malware sample

- Launch **Hex Editor Neo**.
- Open the suspicious file, e.g., `malware.exe`.

🔍 2) Look for Known Headers / File Signatures

Example:

Executables usually start with:

css

CopyEdit

```
4D 5A (MZ) → DOS header
```

-

You might also see:

lua

CopyEdit

```
PE.. (50 45 00 00) → Windows PE format
```

-

This confirms it's a Windows PE binary.

🔍 3) Search for Readable Strings

In Hex Editor Neo:

- Press **Ctrl+F**
- Search for:

    - ASCII
    - Unicode
    - Case insensitive
- Enable "Find all occurrences"

Look for:

Type	Examples
💀 Malware name	`GenericKD`, `AgentTesla`, `njRat`, `DarkComet`, etc.
🔐 Credentials	`user=`, `pass=`, `key=`, etc.
🌐 IPs/Domains	`192.168.`, `.onion`, `.xyz`, etc.
📂 Paths	`C:\Users\...`, `C:\Windows\System32`, etc.
🔖 Attacker nicknames	`By XxH4x0r`, `Coded by R00tKid`, etc.
🧾 Company names	Fake or spoofed like `Microsoft Corp.`

🔍 4) Indicators of Obfuscation or Packing

Indicator	Meaning
Nonsense strings	Could be packed
High entropy	Packed or encrypted
Random or no meaningful strings	Malware uses runtime decryption

You can confirm packing using tools like `PEiD`, `Detect It Easy`, or `binwalk`.

🔍 5) Suspicious URLs or C2 Servers

Search for:

cpp
CopyEdit

```
http://
https://
ftp://
.dll
.php
.asp
```

Malware may try to:

- Connect to command-and-control (C2) servers
- Drop additional payloads
- Send stolen data
-

🔍 6) Company/Certificate/Signer Info (Sometimes Visible)

Search for:

- `Microsoft`
- `Adobe`
- `Google`
  Or any **spoofed name** like `Micros0ft` (typosquatting)

✅ Example Observations

Offset	Content	Meaning
0x0000	`MZ`	Standard Windows executable
0x1234	`By H4ck3rX`	Pseudonym of malware author
0x2450	`http://evilhost.com/bot.php`	C2 server
0x3C00	`Username=admin`	Hardcoded credential
0x4D00	`C:\Users\Admin\AppData\...`	Persistence path

15) 1) Installa and configure snort2) Create a rules set for snort3) Run the snort4) Analyze the result by reading log file.

1) Install and Configure Snort

🖥️ On Linux (Ubuntu/Debian)

bash
CopyEdit

```
sudo apt update
sudo apt install snort -y
```

During installation, it will prompt for:

- **Network interface** to listen on (e.g., `eth0`)
- **Home network IP range** (e.g., `192.168.1.0/24`)

You can also manually set these in `/etc/snort/snort.conf`:

bash
CopyEdit

```
ipvar HOME_NET 192.168.1.0/24
```

✅ 2) Create a Snort Rule Set

🔹 Custom Rule Example

Create a custom rules file:

bash
CopyEdit

```
sudo nano /etc/snort/rules/local.rules
```

Paste a sample rule:

snort
CopyEdit

```
alert icmp any any -> any any (msg:"ICMP Packet Detected";
sid:1000001; rev:1;)
```

16) Open physical build exe file in PEid tool.
🧩 1. Download & Run PEiD
Download PEiD from a reputable source (usually it's a .zip)

Extract and run PEiD.exe (no installation required)

⚠️ Always scan PEiD and its source before use, as it's an older tool and may trigger false positives.

📁 2. Load the EXE File
Click File → Open

Select the suspicious or malware .exe file (e.g., malicious.exe)

🔍 3. Analyze the Output
After loading the file, PEiD will show the following in the main window:

Field	Description
EP Section	Entry Point section of the executable
EP Offset	Offset in file where execution begins
File Offset	Actual file offset

Compiler/Packer	Identified packer, cryptor, or compiler
Entry Point	Often used to detect obfuscation or stubs

🔬 Example Output

yaml

Copy

Edit

```
EP Section: .UPX0
EP Offset: 00001000
File Offset: 00000400
Compiler: UPX v3.02 compressed
```

➡️ This indicates the EXE is packed with UPX (a common executable packer), possibly to evade detection or analysis.

🛠️ 4. Next Steps After Detection

✅ If Packed:

Use unpacking tools like:

upx -d malicious.exe (for UPX)

OllyDbg or x64dbg (for custom/uncommon packers)

✅ If Not Packed:

Proceed with:

Hex analysis (Hex Editor Neo, WinHex)

Static analysis (Dependency Walker, CFF Explorer)

Dynamic analysis (run in a sandbox or VM)

17) 1) Run wireshark with active interface2) Type “http” in the filter and analyze each request carefully.3) Identifiy suspicous URL requests.4) Send those URL to virustotal.com in two form a. Give homepage of the URL b. Give the exact location of the URL taken from wireshark5) Analyze the result.

🔍 1) Run Wireshark with Active Interface

1. Launch **Wireshark**
2. Select your **active network interface** (e.g., Ethernet, Wi-Fi)
3. Start capturing

🔎 2) Apply HTTP Filter

📌 **Filter:**

```
http
```

This will isolate all HTTP traffic (excluding HTTPS).

🧠 3) Analyze HTTP Requests

Look at **each HTTP GET or POST request** in the Info column.

Key Fields to Review:

- **Host**: The domain name
- **Request URI**: The exact path accessed
- **User-Agent**: See if any unusual script or bot is involved
- **Referer**: Who sent the request

➡️ In the **Packet Details Panel**, expand:

```
Hypertext Transfer Protocol
```

Then look for:

- `Host:`
- `Request URI:`
- `Full request URI:` (may not appear directly—build it manually: `http://<Host><Request-URI>`)

🚩 4) Identify Suspicious URL Requests

Look for:

- Unknown domains (e.g., `abc.ddns.net`)
- Long/random strings in URI (e.g., `/images/upload.php?file=abc123.jpg`)
- Background POST requests to domains
- URLs using `.php`, `.exe`, `.jpg` uploads
- Suspicious file extensions or redirects

🔗 5) Scan the URLs on VirusTotal

Go to: https://www.virustotal.com/gui/home/url

⬅️ Submit Both:

a) Homepage of the URL:

Just the domain:

http://maliciousdomain.com

b) Full suspicious path:

http://maliciousdomain.com/upload/screenshot.php?id=123456

> ⚠️ Do NOT click the URLs, just paste them into VirusTotal search bar.

📊 6) Analyze VirusTotal Results

Look for:

- **Detection ratio** (e.g., 12/70 engines marked it malicious)
- **Tags**: phishing, malware, trojan
- **File downloads or redirections**
- **Community comments**

📚 Example

Detected in Wireshark:

Host: example.badactor.com
Request URI: /screenshots/system.jpg

Constructed URL:

http://example.badactor.com/screenshots/system.jpg

Submit to VirusTotal:

- http://example.badactor.com
- http://example.badactor.com/screenshots/system.jpg