9

Troubleshooting Common Issues

In an ideal world, getting up and running with **Microsoft Defender for Endpoint** (**MDE**) is as simple as following the onboarding instructions and checking if machines are reporting in. There are many possible interactions between any security solution and your environment – including between security solutions themselves!

Adding an endpoint security solution to an existing environment will have performance trade-offs, and some tweaking and tuning are required for optimal integration. Things get more complex if you already have security tools of a similar nature running in parallel or if you are migrating from one to another.

Even when carefully following our suggested approach to introducing Microsoft Defender for Endpoint in your environment, the following are some common areas where you will need to apply troubleshooting skills:

- Ensuring the health of the operating system
- Checking connectivity
- Overcoming onboarding issues
- Resolving policy enablement
- Addressing system performance issues
- Navigating exclusion types to resolve conflicting products
- Understanding your update sources
- Comparing files
- Bonus – troubleshooting book recommendations

In this chapter, we will discuss some of the troubleshooting techniques and tools you can use in your environment to identify and solve common issues in these areas.

COLD SNACK

Troubleshooting gets very technical, very quickly. Prerequisite knowledge of systems administration and networking will come in handy; nevertheless, familiarizing yourself with troubleshooting tools and options will at the very least allow for a smoother support case if that turns out to be required. When in doubt, always start with the client analyzer tool!

Ensuring the health of the operating system

The configuration of the operating system can have quite some impact on operations, particularly on Windows, where Defender for Endpoint is tightly integrated. So, it pays to make sure your machine is fully updated and running optimally before proceeding to troubleshoot further. Both the Microsoft **Defender Antivirus** (**Defender Antivirus**) and **Endpoint Detection and Response** (**EDR**) components require regular updates to fix bugs, including performance and stability-related ones, so let's dive into some useful commands to check the update status of the operating system.

Windows

Here are some useful checks to perform:

- Check that there is no filesystem corruption. While you can often see events in the event log that could indicate issues with the disk, checking this one off will help narrow things down:

```
Chkdsk.exe /f c:
```

- Check that there is no OS binary corruption. This command will make sure that operating system files are in a healthy state:

```
DISM /Online /Cleanup-Image /RestoreHealth
```

- Check for the **latest cumulative-update** (**LCU**) using Windows Update.
- Check for the latest **servicing stack update** (**SSU**).
- Update MDE components (for more information, see ***Chapter 7***, *Managing and Maintaining the Security Posture*).

To quickly review what KBs (updates) are installed on a computer, do the following:

1. Open an elevated PowerShell session (**Run as Administrator**).
2. Type in `Get-Hotfix | clip` and press the *Enter* key.
3. Now, type in `notepad.exe` and press the *Enter* key.
4. To paste the information, press *Ctrl* + *V*.

Linux

For Linux, ensuring you are running a supported distribution and kernel version and making sure all the prerequisite packages are installed are the key considerations. Then, you also want to make sure that your repositories are set up successfully. Installation failures typically stem from missing dependencies or heavily modified configurations that disallow installation.

macOS

Newer versions of macOS greatly reduce the matrix of possibilities when it comes to prerequisites. Aside from running the latest and greatest, ensuring no other antimalware solutions are active increases the chances of success. Having a proper (MDM) management solution in place is another critical piece.

A healthy operating system is an important part of ensuring the proper operation of MDE.

Checking connectivity

Connectivity to MDE cloud services is another crucial component. Check the MDE URL list, and make sure that your Windows/Windows Server/macOS/Linux system can connect to the relevant cloud services.

The MDE URL list for commercial customers is located at **https://aka.ms/MDEURL** and provides the most up-to-date reference to which destinations need to be reachable from your machines. In this section, we'll cover a few checks you can perform to check connectivity.

Connectivity quick checks and common issues

To perform some quick spot checks without downloading and running tools, you can use the following commands:

- Check Microsoft Defender Antivirus (Defender Antivirus) Cloud Protection connectivity on Windows:

```
"%ProgramFiles%\Windows Defender\MpCmdRun.exe" -ValidateMapsConnection
```

- Check connectivity to MDE URLs on macOS and Linux:

```
mdatp connectivity test
```

The built-in tools will allow you to check if there are no impairments to connectivity.

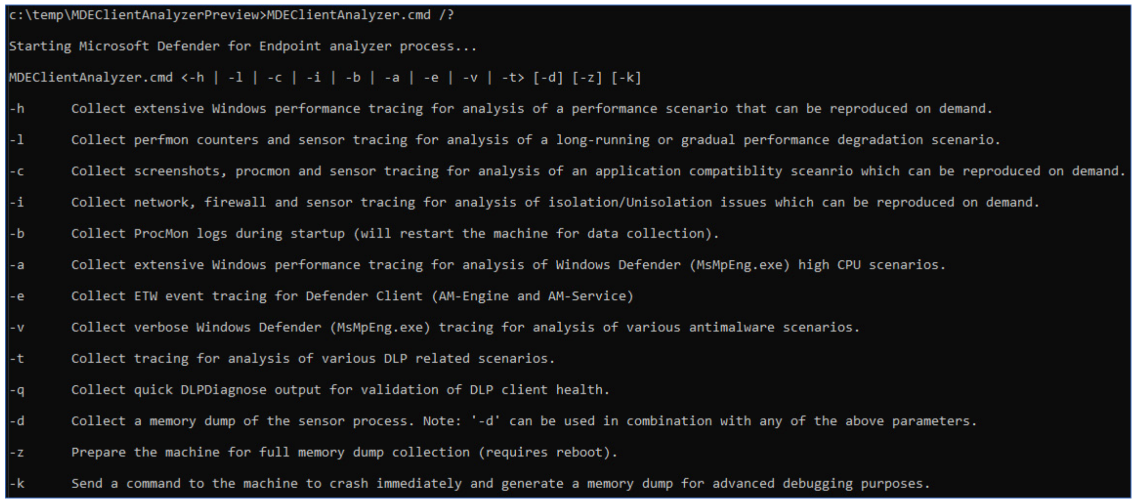
Common issues that can surface are the following:

- There is a proxy that sits between the device and the internet, and the machine is not configured to leverage the proxy
- The proxy or firewall does not allow the connection to the destination at all
- The proxy or firewall does not allow the connection initiated by the specific machine
- The proxy or firewall is set up for user authentication, and since the connection originates from the machine context (**Local System** account), it's not allowed
- The proxy or firewall device is attempting to decrypt (inspect) HTTPS (SSL/TLS) traffic, which fails due to the client using certificate pinning (a security feature to prevent man-in-the-middle attacks)

Once you have ensured that the network and proxy configuration on your machines is in working order, and you have configured your proxy server or firewall to allow traffic to MDE cloud services, your go-to troubleshooting tool will be the client analyzer.

Client analyzer

The client analyzer tool is available from **https://aka.ms/MDEAnalyzer** for Windows and **https://aka.ms/XMDEClientAnalyzer** for Linux and macOS. Note that this tool is very versatile and allows you to collect a lot of diagnostic data, including logs – it's not just limited to network connectivity. It's not often discussed, but the client analyzer (`.cmd`) can be run with several different switches that do a wide variety of troubleshooting and data collection. See the following screenshot for the current list. You can also check out **https://aka.ms/MDEAnalyzerSwitches** for the longer format explanations of each one:



```
c:\temp\MDEClientAnalyzerPreview>MDEClientAnalyzer.cmd /?

Starting Microsoft Defender for Endpoint analyzer process...

MDEClientAnalyzer.cmd <-h | -l | -c | -i | -b | -a | -e | -v | -t> [-d] [-z] [-k]

-h      Collect extensive Windows performance tracing for analysis of a performance scenario that can be reproduced on demand.

-l      Collect perfmon counters and sensor tracing for analysis of a long-running or gradual performance degradation scenario.

-c      Collect screenshots, procmon and sensor tracing for analysis of an application compatiblity sceanrio which can be reproduced on demand.

-i      Collect network, firewall and sensor tracing for analysis of isolation/Unisolation issues which can be reproduced on demand.

-b      Collect ProcMon logs during startup (will restart the machine for data collection).

-a      Collect extensive Windows performance tracing for analysis of Windows Defender (MsMpEng.exe) high CPU scenarios.

-e      Collect ETW event tracing for Defender Client (AM-Engine and AM-Service)

-v      Collect verbose Windows Defender (MsMpEng.exe) tracing for analysis of various antimalware scenarios.

-t      Collect tracing for analysis of various DLP related scenarios.

-q      Collect quick DLPDiagnose output for validation of DLP client health.

-d      Collect a memory dump of the sensor process. Note: '-d' can be used in combination with any of the above parameters.

-z      Prepare the machine for full memory dump collection (requires reboot).

-k      Send a command to the machine to crash immediately and generate a memory dump for advanced debugging purposes.
```

Figure 9.1 – MDE client analyzer switch options

COLD SNACK

This tool should be your go-to! Regardless of whether you intend to open a support case, the tool collects a variety of relevant information, so you don’t have to go and chase down various logs and other sources.

For connectivity checking, the client analyzer tool will attempt to connect to MDE services using the connectivity options available to the device. Note that these checks use Sysinternals’ **`psexec`**, so this needs to be allowed to run in your environment. This makes it a lot easier to test connectivity as it will test all possible destinations using the current (proxy) configuration.

Another benefit of using this tool is that the output is useful in case you need to open a support ticket.

Capturing network packets using Netmon

In case the connectivity analyzer is telling you there is no connectivity issue from the perspective of the endpoint, you may want to dig a little (or a lot) deeper to figure out if there is something else going on with the traffic going toward MDE.

Network Monitor, or **Netmon**, is a very helpful tool to collect the raw packets as they pass through your network and/or wireless adapter. It can be used to diagnose the various network issues you may face.

The most common issues you are likely to encounter in an enterprise are with firewalls (TLS inspection, also known as SSL inspection), proxy servers, and/or **network load balancers** (**NLBs**).

You looked at the event log, you looked at the application log, you tried to check if a port was working, you ran a procmon (or `wprui`), and still can't find what's happening with the application and/or the service.

Netmon can show you the raw packets and decode them to see what data is being passed.

COLD SNACK

A good recommendation when performing this would be to do it without any filters to start; this way, you're not missing anything. As you do this more often, you can get more explicit in your captures. Using Netmon can be more useful than other tools as it gives you the **Process ID** *(***PID***), which can make filtering easier, and we will use it for our example.*

Preparing for a trace

Before you proceed, you want to make sure you are ready to get a clean trace. You can use your favorite network trace capture tool, but you will want to consider gathering at least the following data when troubleshooting a network-related issue:

- IPv4 address
- IPv6 address
- Time (HH:MM:SS)
- IPv4 address of the target
- IPv6 address of the target

- An application that is having the problem, and its PID from Task Manager
- The network share name that is having the problem
- Website name (if troubleshooting a website/web page-related problem)
- Document name (`.doc`, `.docx`, `.xls`, `.xlsx`, and so on)
- Domain name/username (if troubleshooting an authentication problem)
- Domain controller IPv4/IPv6 address
- DHCP IPv4/IPv6 address
- DNS IPv4/IPv6 address

Before starting the trace, perform the following steps:

1. **Minimize the noise**: Close all the applications that are unnecessary for the issue that you are investigating.
2. **Clear any caching that has been done**: Clear your name resolution cache, as well as your **Kerberos** cache:
   1. Clear the DNS name cache:

      ```
      IPConfig /FlushDNS
      ```

   2. Clear the NetBIOS name cache:

      ```
      NBTStat -R
      ```

   3. Clear the Kerberos tickets:

      ```
      KList purge
      ```

3. Next, download and install the latest version of Network Monitor 3.4 (Microsoft Network Monitor 3.4 (archive): https://www.microsoft.com/en-us/download/details.aspx?id=4865).
4. Get ready to reproduce the problem.

For example, let's say you want to capture the **Log Analytics Agent** (**LA**, also known as the **Operations Management Services** (**OMS**) agent, and before that **Microsoft Monitoring Agent** (**MMA**)) network traffic:

1. Open **Control Panel**.
2. Double-click on **Microsoft Monitoring Agent**.
3. Click on the **Azure Log Analytics (OMS)** tab.
4. Open `services.msc` and find the **Microsoft Monitoring Agent** service. Don't do anything yet. You are just staging for reproduction.
5. Open a second CMD (run as admin), type `ping 127.0.0.1`, and do not press *Enter* yet; just before you stop the trace, you will hit *Enter* to create a visual marker for the end of the reproduction.

5. Launch Netmon from the *Start* menu by going to **All Programs** | **Microsoft Network Monitor 3.4**.

Right-click on **Microsoft Network Monitor 3.4**. Click on **Run as admin**. If you are prompted with the **Microsoft Update Opt-in** option, click on **No**.

6. Select the appropriate network interface. When you first run Netmon, you will set which network interface to use for the trace:
   1. Under **Select Networks**, check all the boxes (unless you are working on a Hyper-V server, you might want to limit to the network that you are investigating).
   2. The following command should help you identify the appropriate interface via the **Physical Address** (run CMD as admin):

```
ipconfig /all
```

7. Increase the buffer settings.

By default, **`Netmon`** will only trace up to 20 MB of data before it starts to overwrite the capture buffer. Set the buffer to a larger size (say 1 GB):

1. Click on **Tools** | **Options...** | **Capture**.
2. Under **Temporary capture file**, select **Size: 1024 Megabytes** and click on **OK**.

Now that you're all set up and ready to reproduce, let's start the trace.

Running the trace

Now, we will start the trace and reproduce the issue. In the end, we will flag the end using our ping and save the trace:

1. Click on the **New capture** tab.
2. Click the start button or press *F5*.
3. **Frame Summary** will start populating new frames.
4. Reproduce the issue. For our example, we will restart Microsoft Monitoring Agent (*Step 4*/*B* from the previous section).

NOTE

It takes less than 30 seconds for MMA to communicate properly unless you are having network issues.

5. On the second CMD where you had typed **`ping 127.0.0.1`**, now press *Enter*.
6. Stop the trace by clicking the stop icon in the toolbar or hitting *F7*. For the MMA example, you might want to wait a few minutes since it downloads the **`.cab`** file, extracts it, then communicates with Azure Log Analytics before the service is finally started.
7. Save the trace via **File** | **Save As**. Leave the default radio button selection set to **All captured frames**.

8. If you're sending out your trace for analysis, make sure that you capture network configuration information in a **`.txt`** file. Go to *Start* | CMD (run as admin):

```
ipconfig /all > %computername%-ipconfig.txt
```

```
tasklist >> %computername%-ipconfig.txt
```

```
tasklist /svc >> %computername%-ipconfig.txt
```

Once you've done this, zip up the **`.cap`** and **`.txt`** files to send them out.

Viewing traces

To view your traces, launch **`NetMon.exe`**, choose the **File**/**Open**/**Capture** menu, and open the **`.cap`** file. When you open a trace file, you will see that **`NetMon.exe`** displays the traces at various layers.

What are we looking for? Well, since the goal here is to identify connectivity issues between the MDE client (processes) and cloud services, this is what we would focus on. Most commonly, you are looking for denied or unsuccessful connections that involve one of the MDE processes and destinations. Here are two filters you can start with:

- **Protocol**: All MDE connections are HTTPS – the only exceptions could be connections to certificate-related services (certificate revocation lists)
- **Processes**: You can look up MDE processes as well as destinations at **https://aka.ms/MDEURL**

Common indicators of connection issues are dropped packets and resets (`RST`). When a proxy is involved, you may want to look for `CONNECT` as it indicates the start of a session.

TIP

You can automate stopping a trace based on, for example, an event that fires, which is very useful when you're trying to capture an intermittent issue that's hard to manually reproduce. See How to Stop a Network Trace Programmatically using Network Monitor at **https://techcommunity.microsoft.com/t5/core-infrastructure-and-security/fire-amp-forget-how-to-stop-a-network-trace-programmatically/ba-p/256200** *for more details.*

Here are some other useful network packet capture tools:

- **https://www.telerik.com/download/fiddler**
- **https://www.wireshark.org/download.html**

Ensuring proper connectivity is an important piece to ensuring onboarding success. Read on for troubleshooting options during the onboarding phase.

Overcoming onboarding issues

This section describes common onboarding issues and how to overcome them. On Windows, your go-to for a spot check would be the `SENSE` operational log in Event Viewer. That said, the most common reasons for onboarding issues are the following:

- Connectivity issues
- Missing prerequisites for installation

Again, the **MDE Analyzer script** is extremely useful for surfacing common issues and gathering logs, so it will likely be your best first step. With that said, let's cover troubleshooting the various types of onboarding.

Troubleshooting onboarding issues

In general, there are one or two distinct steps when onboarding a machine to MDE:

1. Installation
2. Registering the device to connect to your tenant

Both steps are critical. Installation typically requires meeting the prerequisites. Troubleshooting this step mostly comes down to checking the installer output, which can be done visually, on the command line, or in the installation logs. For an overview of log and configuration locations, please see ***Chapter 10***, *Reference Guide, Tips, and Tricks*.

When it comes to registering your machine to your tenant, this is typically performed by a script or via a tool that configures the device to connect to the right tenant and register with the right *secret* (key). Most of the time, connectivity is a critical piece here.

To understand what could go wrong during the registration part, it's good to know what the device is doing. The following is a high-level overview:

1. After installation, you apply a configuration to the device, which includes the secret piece and where the device should report to.
2. At the end of the configuration, the EDR services are started.
3. The service responsible for sending telemetry starts communicating with the cloud service, kicking off the authentication and registration process.
4. The cloud service sends a configuration file down to the client.
5. The client is now configured to collect and send telemetry and will send a full machine report.

Here is a list of quick checks:

- Has the EDR process successfully started? If not, the onboarding script may have failed.
- To check if the onboarding blob was applied successfully, check the registry (Windows) or the configuration file (Linux and macOS). Check if the **orgID** matches your tenant (you can find this in the `security.microsoft.com` portal) to make sure the onboarding information is correct for your tenant.
- If it was applied successfully but the service hasn't started, check the system event logs to see if something prevented a successful start.
- If everything is running as expected but you don't see the machine object in the portal even after waiting a while, you may have a connectivity issue.

With some of the modern onboarding checks covered, let's take a look at how the MMA agent differs from the newer unified agent.

MMA versus the new unified agent

As you learned in ***Chapter 6***, *Considerations for Deployment and Configuration*, for Windows Server 2012 R2 and 2016, Microsoft launched a revamped agent in 2022. There are a few key differences with the previous solution, and some things to be aware of if you're troubleshooting onboarding issues:

- **Connectivity**: The connectivity requirements are an exact match with Windows Server 2019. The previous MMA-based agent for older OSs could connect through an **Operations Management Services** (**OMS**) gateway; this isn't the case for the unified agent.
- **Prerequisites**: All prerequisites can be met by ensuring the machines you are deploying MDE to are fully up to date. The easiest way to take care of this is by installing the LCU, which likely requires the latest SSU. While you can attempt to meet the bare minimum requirements, the reality is that you are introducing risks and variables when attempting to secure an unpatched system.

- **Defender Antivirus feature (Windows Server 2016)**: On Windows Server 2016, the Defender Antivirus feature must be activated and updated all the way. Some third-party antimalware solutions actively disable Defender Antivirus, so it may take some extra steps to recover from that. Use the `mpcmdrun.exe /wdenable` command to ensure it starts, and make sure to update using the latest platform update before proceeding with the installation.
- **Installation logs**: If you're running the MSI installer, using `/V` will provide the most verbose logs. However, the best way to troubleshoot installation issues is to run the installation helper script.

COLD SNACK

Microsoft has released an installation helper script. If you're not using Microsoft Defender for Cloud or ConfigManager to automate the deployment, this script is very useful to check for and resolve prerequisites before attempting to install. It will also allow you to orchestrate the execution of the onboarding script – and it generates useful verbose output and logs in case anything goes wrong.

Custom indicators

Indicators have several prerequisites, and you need to meet all of them (see ***Chapter 10***, *Reference Guide, Tips, and Tricks – Interdependent settings* for more information):

- Defender Antivirus must be in active mode with real-time protection and cloud-delivered protection enabled for file indicators to work.
- For network indicators, you also need to enable network protection in block mode.
- You must configure your tenant in the `security.microsoft.com` portal to enable the allow or block files and custom network indicators functionality. For more information about tenant configuration, see ***Chapter 6***, *Considerations for Deployment and Configuration.*

Web content filtering

Web content filtering (**WCF**) requires the following to work:

- An active content filtering policy.
- The Edge browser or Chrome, Firefox, Brave, or Opera and the **network protection** feature enabled in block mode with **custom network indicators** turned on in the portal. On servers, only Edge is supported.

If you have everything in place and are still seeing that certain websites are allowed, you may want to consider the precedence order (MDCA policy refers to unsanctioned web applications from the Microsoft Defender for Cloud Apps integration). The following table shows the order of evaluation and the resulting action:

Custom IoC Policy	**Web Threat Policy**	**WCF Policy**	**MDCA Policy**	**Result**
Allow	Block	Block	Block	Allow (Web Protection Override)
Allow	Allow	Block	Block	Allow (WCF Exception)
Warn	Block	Block	Block	Warn (Override)
No Policy	Allow	Block	Sanctioned (Allow)	Block (MDCA Allow does not generate indicators)

Table 9.1 – WCF and custom IoC precedence order

After successful onboarding and testing indicators, like with any security solution, you may run into configuration challenges. The following section suggests various ways of troubleshooting issues in this category.

Resolving policy enablement

When first deploying MDAV settings, in either a PoC, test, or production environment, you might run into a challenge due to conflicting policies. There are various ways to find out where settings are coming from.

The traditional method of investigating policy conflicts is by using **`gpresult.exe`**. The following command applies to group policies:

```
Gpresult.exe -h > c:\temp\GPResult_output.html
```

You will get an HTML report that will tell you which settings are effective – and which policies are the sources of these settings.

COLD SNACK

Remember that group policies have a processing order – the local policy is the first to be evaluated and the further away (OU, then domain) the additional policies are, the higher their precedence. The last setting applied wins.

For Intune/MDM, the tool of choice is the MDM diagnostic tool:

```
mdmdiagnosticstool.exe -out c:\temp
```

A file named `MDMDiagReport.html` will be created in the specified directory.

You will want to look for the Defender CSP, as well as `ADMX_MicrosoftDefenderAntivirus`, to determine which settings were applied.

Alternatively, use **Settings** | **Home** | **Accounts** | **Access work or school** | **Info** | **Create report**.

The file will be located in `c:\users\public\public Documents\MDMDiagnostics\MDMDiagReport.html`.

Checking settings

You can use the PowerShell `Get-MpComputerStatus` and `Get-MpPreference` commands to check which settings are effective. For a more detailed overview, see ***Chapter 10***, *Reference Guide, Tips, and Tricks*.

These are the registry locations for Defender Antivirus configurations:

Source	**Registry location**
Domain GPO/ Local GPO	`HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender`
MDM (Intune)	`HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender\Policy Manager`
PowerShell (`Set-MpPreference`)	`HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows Defender`

Table 9.2 – Defender Antivirus configuration registry location

Precedence order

It's important to understand, if only to be able to troubleshoot, what the precedence order is that Defender maintains in the respective settings coming from different sources.

COLD SNACK

In general, local preferences get overruled by any policy, as specified via management channels. When **tamper protection** *(***TP***) is also applied, certain settings (but not all) are protected from modification.*

The following sections go over the precedence order of the various configuration channels per operating system.

Windows

The order is as follows: preferences | MDE security configuration management | **mobile device management** (**MDM**) | group policy. To be more specific:

- Group policy wins over all other channels. However, there is an evaluation order inside the group policy: the last applied setting wins. In a domain, this means local policy loses out to Organizational Unit policy which gets overridden by domain policy in turn (for those settings that have been defined!) (`gpresult.exe` can tell the story).
- MDE security configuration management uses the policy hive, so it essentially sits on the same level as a local group policy. However, it will yield (stop working) if *any* MDM is managing the device.
- MDM (CSP) is next on the list.
- Preferences, typically configured using PowerShell (or the user interface), are last.

macOS

The local configuration (Terminal) gets overridden by managed the MDM configuration (Intune, JAMF, AirWatch, and so on).

Linux

The local configuration (bash) gets overridden by the managed configuration file (JSON).

After ensuring the right configuration has been applied successfully, you may need to understand system performance issues – and identify opportunities for improvement.

Addressing system performance issues

Performance impact is likely the biggest reason you may need to troubleshoot. While the outcome of this may be an exception of sorts, understanding the actual issue will help you to precisely scope this exception.

One useful capability that will buy you some time is **troubleshooting mode**. This mode doesn't perform troubleshooting for you; it temporarily allows you to turn off the **tamper protection** feature so that you can locally make modifications to the configuration. This provides you with quick mitigation (turn something off), but more importantly, it provides you with the flexibility to troubleshoot further.

Here's some information you will want to have at hand before you start:

- Is it impacting one or more systems?
- Is it impacting one end user or multiple end users?
- How do they recover? Close apps (kill apps)? Reboot?
- How much RAM does the system have?
- How many CPU cores are in the system?

The answers to these questions will help you identify if you need to act broadly, such as reverting a setting or a patch, or if you need to dig deeper on a specific machine.

Windows

In Windows, particularly modern Windows operating systems, you will find several built-in tools to help troubleshoot performance issues, and some optional ones as well. We will cover a few different options; depending on your skills and desire to understand more about performance impact, you may choose one over the other.

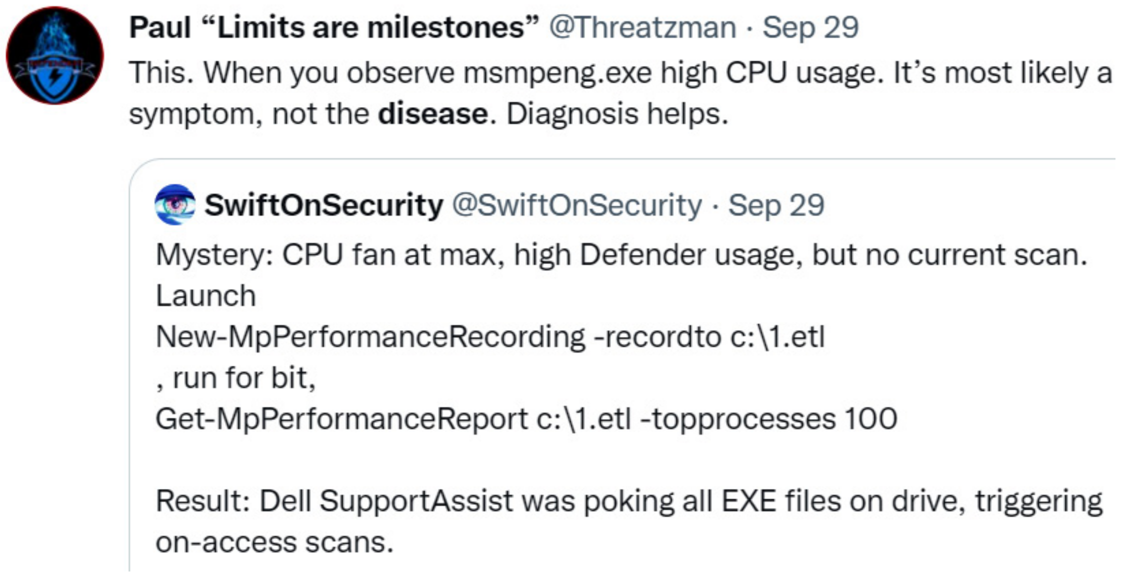
MDAV Performance Analyzer

In 2021, Microsoft shipped the **performance analyzer** tool for Microsoft Defender Antivirus. This tool combines a variety of inputs, captures a trace, and provides you with a report of its findings.

It's arguably the best tool to quickly find out which piece of software is causing MDAV to respond. It automates the process of manually parsing logs; it can capture a trace and gives you quick insights.

Alternatively, you can manually check the Microsoft protection log file in `C:\ProgramData\Microsoft\Windows Defender\Support`. In `MPLog-xxxxxxxx-xxxxxx.log`, you can find entries that indicate how much time was spent scanning a specific process.

Note that the MDAV performance analyzer is for Defender Antivirus only. It's a great first step. Also, remember the following when you see high CPU usage for `msmpeng.exe` (the MDAV antimalware process):


Paul “Limits are milestones” @Threatzman · Sep 29
This. When you observe msmpeng.exe high CPU usage. It’s most likely a symptom, not the **disease**. Diagnosis helps.

SwiftOnSecurity @SwiftOnSecurity · Sep 29
Mystery: CPU fan at max, high Defender usage, but no current scan.
Launch
New-MpPerformanceRecording -recordto c:\1.etl
, run for bit,
Get-MpPerformanceReport c:\1.etl -topprocesses 100

Result: Dell SupportAssist was poking all EXE files on drive, triggering on-access scans.

Figure 9.2 – Example of performance troubleshooting

Using Performance Monitor (PerfMon) to find issues involving MsSense.exe

The built-in Performance Monitor tool can help you to understand what’s going on with the other main component of MDE, `MsSense.exe`, and whether you need to find out if another factor may be the dominant one at the source of the issue.

This section talks about which metrics are important to track when investigating performance impact using PerfMon.

COLD SNACK

*You may want to start your performance troubleshooting session with process monitor (***procmon***) to capture a trace when the issue occurs. This will allow you to zoom in on which processes appear to cause the impact, helping to narrow down further troubleshooting steps.*

Interrupt Service Routines (ISRs) and Deferred Procedure Calls (DPCs)

Interrupt Service Routines (**ISRs**) and **Deferred Procedure Calls** (**DPCs**) are what are known as **silent performance killers**.

These are so important that Microsoft added the ISR and DPC information to Task Manager. An ISR is a driver. It's a driver of a physical device that receives interrupts – it will register one or more interrupt service routines to service these interrupts.

The following can be used to help determine if there are issues to zoom in on in this area:

- The **`MSSense.exe`** mini-filter driver (**`MSSecFlt`**) is optimized where there is no noticeable ISR overhead
- The **`MSSense.exe`** mini-filter driver (**`MSSecFlt`**) is optimized where there is no noticeable DPC overhead
- **% privileged time** in **PerfMon** is a calculation of DPC time, kernel time, and interrupt (ISR) time and indicates overhead and should typically have a low (1%) average
- The thresholds differ for ISRs and DPCs – network I/O can get impacted at 15% and higher, and disk I/O can get impacted at 5% and higher

Processor

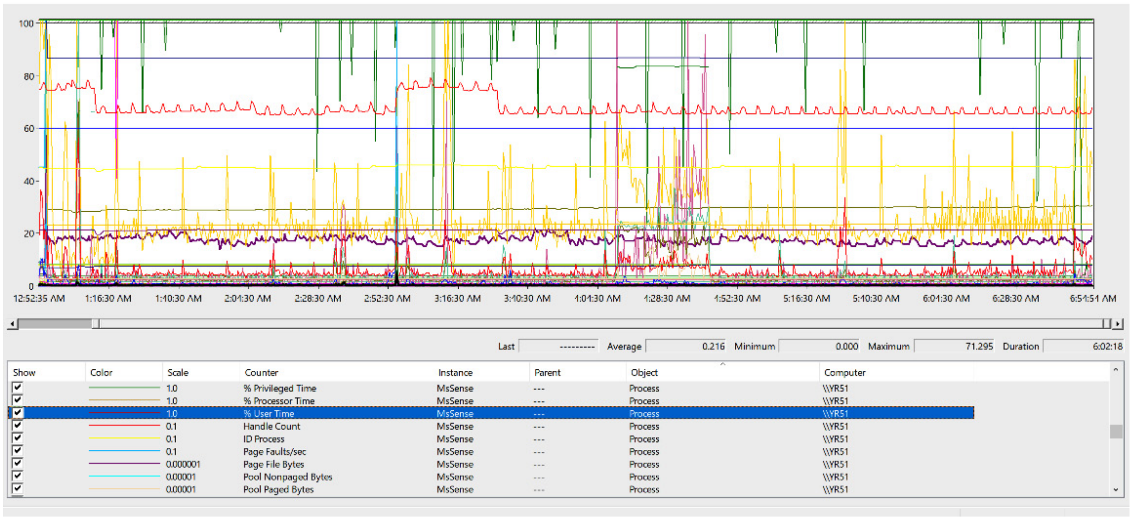
When it comes to processor time (the amount of capacity allocated for a given duration), what's important is to understand the duration of a spike and the number of processors in the machine. Spiking to 100% seems bad, but unless it's only for a brief period, it might not be indicative of a problem. Similarly, on a multi-core system, using 100% of a single core is less impactful than using multiple cores at a high load, especially if the thread priority of the process is normal.

Check the following list to help find problems:

- Use the **% Processor Time** counter to find out the duration of the spike. A short spike is likely OK, as is 100% consumption of a single core during that spike.

- Check the thread priority of the process in Task Manager. If the priority is normal, this means that the process is not likely to take time away from other processes. You can use the **Metric Priority Base** counter to measure; 8 means normal priority.
- **% User Time** indicates the amount of time spent in user mode.
- The **System\Context Switches/sec** counter is used to report system-wide context switches.
- The **Thread(_Total)\Context Switches/sec** counter is used to report the total number of context switches, generated per second – by all threads.

Here's what PerfMon may look like once you add some common processor and memory-related counters:



Last --------- Average 0.216 Minimum 0.000 Maximum 71.295 Duration 6:02:18

Show	Color	Scale	Counter	Instance	Parent	Object	Computer
✓	🟩	1.0	% Privileged Time	MsSense	---	Process	\\YR51
✓	🟫	1.0	% Processor Time	MsSense	---	Process	\\YR51
✓	🟥	1.0	% User Time	MsSense	---	Process	\\YR51
✓	🟥	0.1	Handle Count	MsSense	---	Process	\\YR51
✓	🟨	0.1	ID Process	MsSense	---	Process	\\YR51
✓	🟦	0.1	Page Faults/sec	MsSense	---	Process	\\YR51
✓	🟪	0.000001	Page File Bytes	MsSense	---	Process	\\YR51
✓	🟦	0.00001	Pool Nonpaged Bytes	MsSense	---	Process	\\YR51
✓	🟨	0.00001	Pool Paged Bytes	MsSense	---	Process	\\YR51

Figure 9.3 – PerfMon with the processor and memory-related counters active

COLD SNACK

If you are running a machine on any virtualization platform, chances are that physical cores are shared by multiple machines. If there are too many machines claiming processor time, the physical CPU can get overloaded by context switches – multiple virtual machines fight to get their threads executed, the CPU usage starts spiking, and overall performance is drastically impacted for all machines. While context switching itself doesn't generate much overhead, it's a clear sign of oversubscription.

Memory

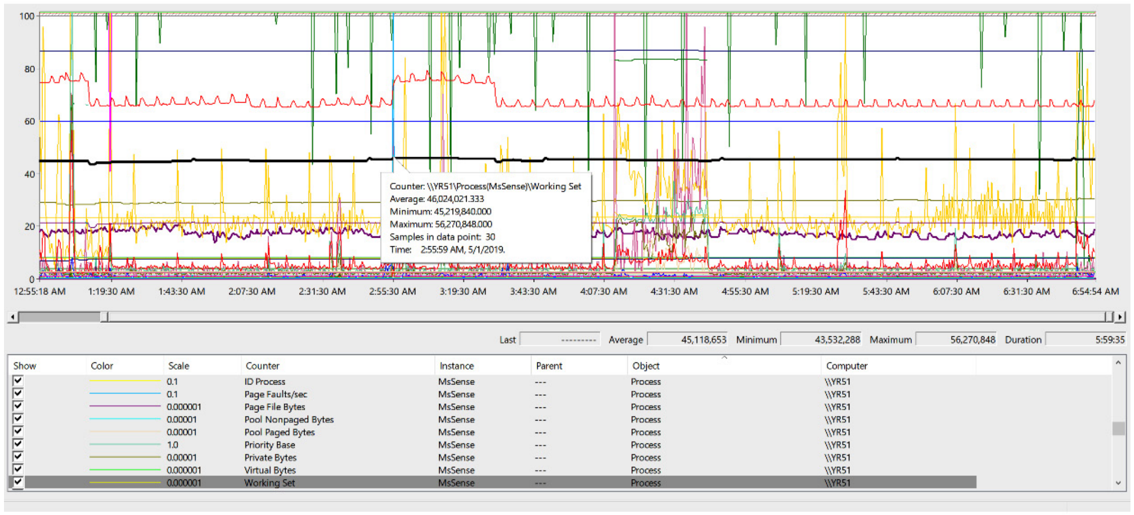
Memory is another important aspect of the overall performance of the machine. Issues here can also be exacerbated by disk performance during low memory conditions – in turn, when memory is low processing performance is also impacted.

Something that's particularly critical is kernel memory – what happens when it's exhausted?

- The server/workstation becomes sluggish
- Performance bottlenecks
- The OS may decline additional connections or requests
- Application failures
- Random OS/app errors
- Blue Screen errors

There are two types of memory to track: **paged pool** and **non-paged pool** (kernel memory). Why are they important? Well, if a leak is caused by the mini-filter kernel driver (for the MDE EDR sensor, this would be `MsSecFlt.sys`), it can lead to the system becoming unstable, ultimately leading to the machine hanging or bug-checking (**Blue Screen of Death** or **BSOD**).

To find out the paged pool and non-paged pool usage by `MSSecFlt`, use the **Pool Paged Bytes** and **Pool Nonpaged Bytes** counters, respectively:



Counter: \\YR51\Process(MsSense)\Working Set
Average: 46,024,021.333
Minimum: 45,219,840.000
Maximum: 56,270,848.000
Samples in data point: 30
Time: 2:55:59 AM, 5/1/2019.

Last --------- Average 45,118,653 Minimum 43,532,288 Maximum 56,270,848 Duration 5:59:35

Show	Color	Scale	Counter	Instance	Parent	Object	Computer
✓	🟨	0.1	ID Process	MsSense	---	Process	\\YR51
✓	🟦	0.1	Page Faults/sec	MsSense	---	Process	\\YR51
✓	🟪	0.000001	Page File Bytes	MsSense	---	Process	\\YR51
✓	🟦	0.00001	Pool Nonpaged Bytes	MsSense	---	Process	\\YR51
✓	⬜	0.00001	Pool Paged Bytes	MsSense	---	Process	\\YR51
✓	🟩	1.0	Priority Base	MsSense	---	Process	\\YR51
✓	🟫	0.00001	Private Bytes	MsSense	---	Process	\\YR51
✓	🟩	0.000001	Virtual Bytes	MsSense	---	Process	\\YR51
✓	🟨	0.000001	Working Set	MsSense	---	Process	\\YR51

Figure 9.4 – Example from Performance Monitor

You need to look at things holistically as one thing can lead to another: if you observe a lot of page file usage, this is indicative of a memory shortage. This can lead to high disk I/O, which, in turn, can lead to high CPU usage as operations are queued.

Some of the more useful memory-related counters you want to pay attention to include the following, where any excesses may indicate a problem.

Working Set

A **Working Set** is the size (in bytes) of the *working set* of a process. The counter shows the set of memory pages touched by the threads in the process most recently. If free memory goes above a certain threshold, pages are left in the working set, even if they become unused. When it drops below a certain threshold, pages are trimmed from the working set. If they are needed, they will then be *soft-faulted* back into the working set before leaving the main memory.

Private Bytes

The **Private Bytes** counter shows how much memory (in bytes) the process has set aside and which cannot be given out to other processes.

Page File Bytes

The **Page File Bytes** counter shows the amount of virtual memory (in bytes) that a process has set aside for use in the paging file(s). Paging files store pages of memory used by the process that are not contained in other files. Used by all processes, a shortage of space in paging files can prevent other processes from allocating memory. If there is no paging file, this counter reflects the amount of virtual memory that the process has reserved in physical memory instead.

Virtual Bytes

`MSSense.exe` typically uses a maximum of 2 TB of virtual bytes. All 64-bit apps have a default maximum set at 2 TB. The max available setting is 126 TB.

Virtual Bytes is the current amount of virtual memory, in bytes, that this process has been assigned by the operating system across virtual addresses in physical memory.

Handle Count

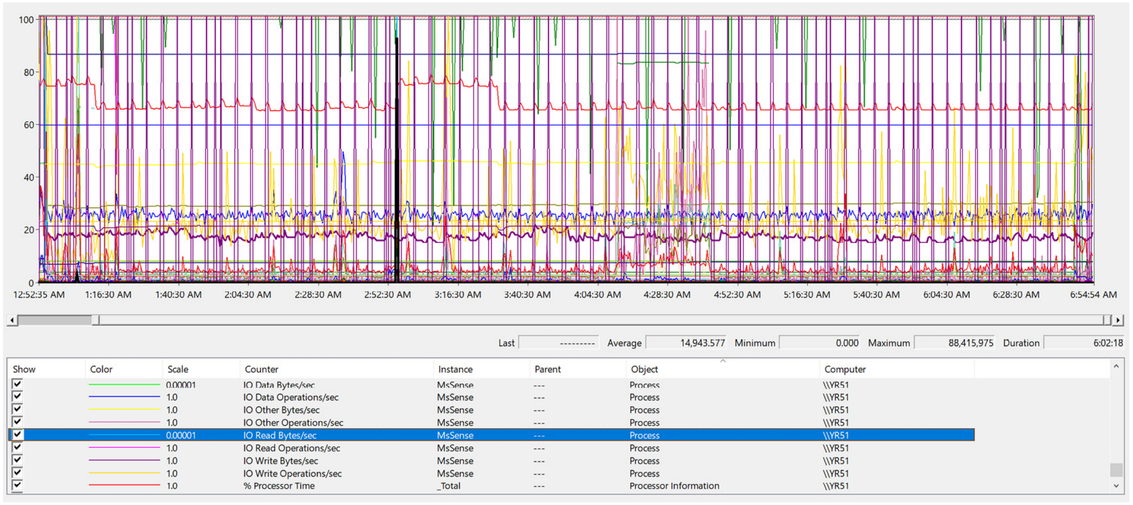
Crossing a threshold of 3,000 handles would indicate a need to start further investigation.

Handle Count is the sum of all handles a process has open – handles are connections to different objects in the operating system, including, but not limited to, files, resources, and memory locations.

Disk I/O, Network I/O, Device I/O

What was `MSSense.exe` doing when there was a CPU spike? PerfMon's **IO Data Bytes/sec** doesn't split the **Disk I/O**, **Network I/O**, or **Device I/O** by process.

It's important to find out if the process you are examining is doing mostly writes or reads to find out if there's a problem. The following screenshot shows an example of this:



Last	Average	Minimum	Maximum	Duration
---------	14,943.577	0.000	88,415,975	6:02:18

Show	Color	Scale	Counter	Instance	Parent	Object	Computer
✓	🟩	0.00001	IO Data Bytes/sec	MsSense	---	Process	\\YR51
✓	🟦	1.0	IO Data Operations/sec	MsSense	---	Process	\\YR51
✓	🟨	1.0	IO Other Bytes/sec	MsSense	---	Process	\\YR51
✓	🟥	1.0	IO Other Operations/sec	MsSense	---	Process	\\YR51
✓	🟦	0.00001	IO Read Bytes/sec	MsSense	---	Process	\\YR51
✓	🟪	1.0	IO Read Operations/sec	MsSense	---	Process	\\YR51
✓	🟪	1.0	IO Write Bytes/sec	MsSense	---	Process	\\YR51
✓	🟨	1.0	IO Write Operations/sec	MsSense	---	Process	\\YR51
✓	🟥	1.0	% Processor Time	_Total	---	Processor Information	\\YR51

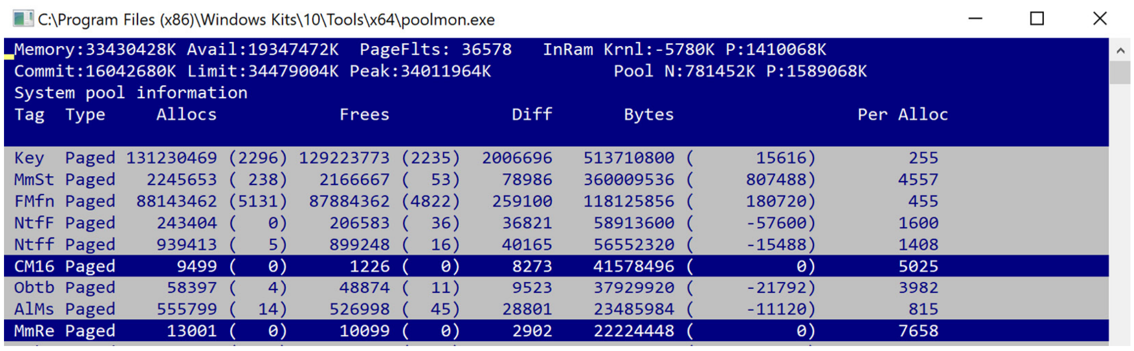
Figure 9.5 – Example from Performance Monitor

Compare **IO Data Bytes/sec** and **IO Read Bytes/sec** to find out how much of the I/O consisted of reads.

Finding a kernel-mode memory leak

The **Windows Driver Kit** (**WDK**) tool known as PoolMon allows you to identify what is using most of the paged pool (kernel memory). You will need to know what pool tag to look for: https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/using-poolmon-to-find-a-kernel-mode-memory-leak.

The following is an example of what you'll see when you launch PoolMon – that is, a quick display of what the OS is collecting about memory allocations:


C:\Program Files (x86)\Windows Kits\10\Tools\x64\poolmon.exe

```
Memory:33430428K Avail:19347472K  PageFlts: 36578   InRam Krnl:-5780K P:1410068K
Commit:16042680K Limit:34479004K Peak:34011964K            Pool N:781452K P:1589068K
System pool information
Tag  Type     Allocs            Frees            Diff        Bytes                 Per Alloc

Key  Paged 131230469 (2296) 129223773 (2235)  2006696  513710800 (     15616)        255
MmSt Paged   2245653 ( 238)   2166667 (  53)    78986  360009536 (    807488)       4557
FMfn Paged  88143462 (5131)  87884362 (4822)   259100  118125856 (    180720)        455
NtfF Paged    243404 (   0)    206583 (  36)    36821   58913600 (    -57600)       1600
Ntff Paged    939413 (   5)    899248 (  16)    40165   56552320 (    -15488)       1408
CM16 Paged      9499 (   0)      1226 (   0)     8273   41578496 (         0)       5025
Obtb Paged     58397 (   4)     48874 (  11)     9523   37929920 (    -21792)       3982
AlMs Paged    555799 (  14)    526998 (  45)    28801   23485984 (    -11120)        815
MmRe Paged     13001 (   0)     10099 (   0)     2902   22224448 (         0)       7658
```

Figure 9.6 – Example from Performance Monitor

Compare the values over time – if a tag is always increasing its allocation (bytes), this is indicative of a memory leak.

Non-paged pool memory (kernel memory)

For non-paged pool memory (kernel memory) usage, you can also use PoolMon. To find the driver that is using the memory, you can leverage the `pool` tag and perform a search in `c:\windows\system32\drivers`:

```
C:\Windows\System32\drivers\findstr /m /m FDRo *.sys
```

As we can see, it turns out that the **FDRo** tag is used by `SysTrace.sys`.

`sysTrace.sys` is used as a part of the Microsoft Software Certification Toolkit, which is unrelated to MDE.

Note that most tools that perform memory leak analysis for applications and services collect two types of user-mode memory when troubleshooting leaks:

- User mode (apps/services) leak = Private Bytes (Heap)
- User mode (apps/services) leak = Virtual Bytes (VAllocs)

DebugDiag will allow you to track the memory allocations. Download the Debug Diagnostic Tool and set it up to track either **Private Bytes** or **Virtual Bytes**.

Capturing performance logs using Windows Performance Recorder (WPR)

A great way to troubleshoot performance is by getting into the nitty gritty – **Windows Performance Recorder** (**WPR**) allows you to capture a trace so that you can analyze it further.

You will need to install **Windows Performance Toolkit** (**WPT**) v5.0 (containing WPR, the user interface, and the Xperf tool) first.

TIP

If you have multiple machines where the issue is reproducing, use the machine with the most amount of RAM. For example, on a Windows client, use a machine with 16 GB of RAM or more if available.

NOTE

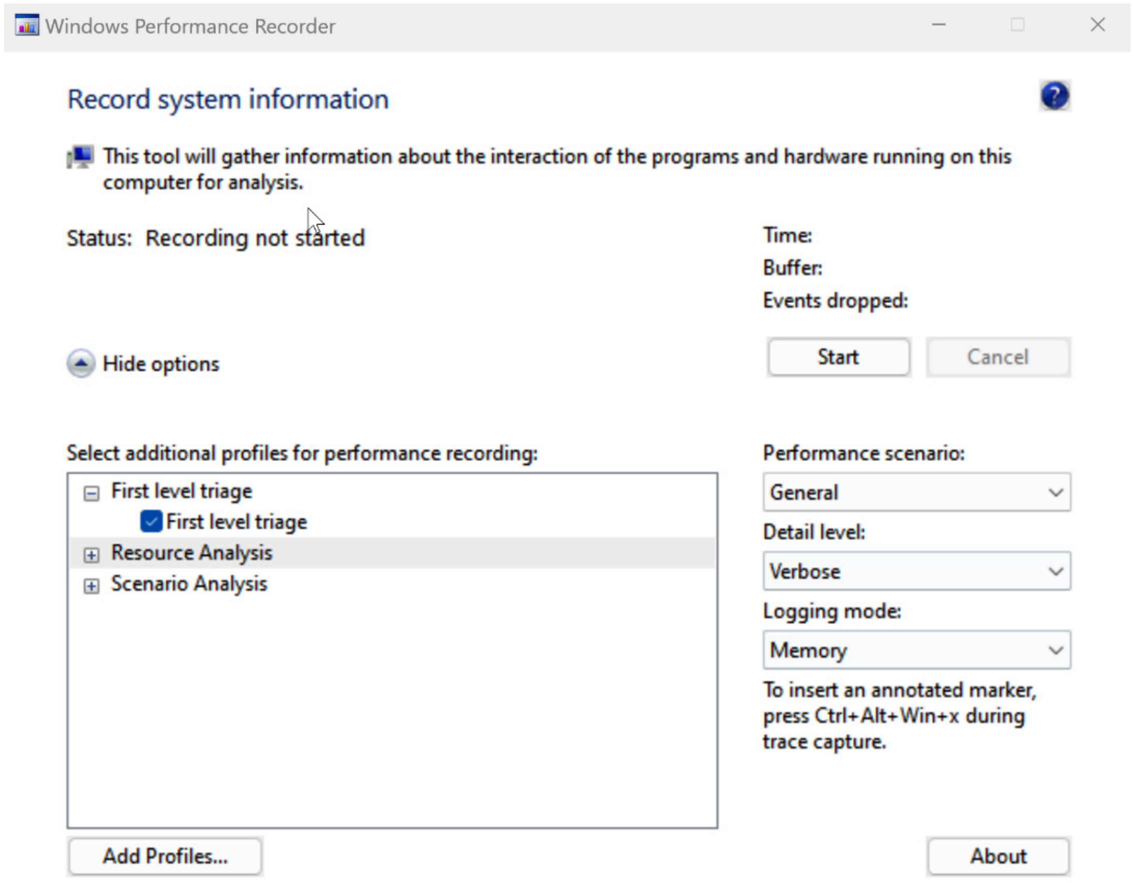
The WPT is available in the Windows 10 **Advanced Deployment Toolkit** *(***ADK***) or* **Software Deployment Kit** *(***SDK***). The SDK also provides Debugging Tools for Windows, which you may find useful.*

The Windows 10 ADK is available at **https://aka.ms/Win10ADK**.

The Windows 10 SDK is available at **https://aka.ms/Win10SDK**.

Setting WPR up to collect data

After installation, some configuration is required. Launch WPR as an administrator and then click on the **More options** dropdown to select profiles:



This tool will gather information about the interaction of the programs and hardware running on this computer for analysis.

Status: Recording not started

Time:
Buffer:
Events dropped:

Select additional profiles for performance recording:

- First level triage
  - [✓] First level triage
- Resource Analysis
- Scenario Analysis

Performance scenario: General
Detail level: Verbose
Logging mode: Memory

To insert an annotated marker, press Ctrl+Alt+Win+x during trace capture.

Figure 9.7 – Performance Monitor settings example

COLD SNACK

On Windows clients, since most machines have less than 16 GB of memory, set **Logging mode** *to* **File** *instead of* **Memory** *to prevent buffer drops.*

IMPORTANT

If your machine has 64 GB of RAM or more, click **Add Profiles** *and select* **`C:\Program Files (x86)\Windows Kits\10\Windows Performance Toolkit\SampleGeneralProfileForLargeServers.wprp`**. *Without this, your computer can use much more non-paged memory (buffers), which can cause the system to become unstable.*

You can select profiles for performance recording. The following table will help you select the right ones for your given scenario:

WPR profile	When to capture	Recommendation

First level triage	Basic capture.	Always collect
CPU usage	This will show high CPU usage in application(s), service(s), or the **System** process. This can show how long applications hang.	Always collect
Disk I/O activity	Can show the application or service that is causing high disk usage, even storage drivers that have high I/O.	Always collect
File I/O activity	Shows the files and folders that are being touched.	Always collect
Registry I/O activity	Here, you can look at registry hits and modifications.	Always collect
Networking I/O activity	Provides the local and target IP addresses, as well as the dynamic and target ports that various applications are using.	Always collect
Heap usage	**Private bytes** (memory leaks in user mode).	N/A
Pool usage	**Paged pool** and/or **Nonpaged pool** (memory leaks in kernel mode) excessive usage.	N/A

VAlloc usage	**Virtual bytes** (memory leaks for user mode).	N/A
Power usage	Changes in power for the processor.	N/A
GPU activity	Performance of the video card.	N/A
Audio glitches	Experiencing stuttering audio on a call.	N/A
Video glitches	Bad video quality.	N/A
Edge Browser	Slowdowns when browsing.	N/A
Minifilter I/O activity	Here, you can find out which security products such as Antivirus, DLP, HIPS, and EDR are slowing you down.	Always collect

Table 9.3 – Profile options in Performance Monitor

For real-life production issues, you will want to select the following after expanding **Resource Analysis**:

- **CPU Usage**
- **Disk I/O activity**
- **File I/O activity**
- **Registry I/O activity**
- Expand **Scenario Analysis**
  - **Minifilter I/O activity**

When troubleshooting MDAV, you can download the sample ETW profile from https://github.com/MDATP/Scripts/blob/master/MDAV_ETW_Profile.wprp, then perform the following two actions:

- Use the **Add Profiles…** button to import your downloaded file. You should be able to see **Custom measurements**. Select `MDAV_ETW_Profiling`.
- Finally, change **Logging mode** from **Memory** to **File**. You are now ready to start recording.

Collecting data

Close all the applications that are not necessary to reproduce the issue. We want to reduce the amount of data that is going to be collected as it will make everyone's lives easier when analyzing the dataset.

Click on **Hide options** so that just the **Start** button is visible. Make sure you have your repro (reproduction of the issue) ready (for example, application, network folder, or other):

1. Click on **Start** and reproduce the issue to generate activity in the log we can review. Keep the data collection to less than 5 minutes – ideally in the 2 to 3 minutes range – since there is a lot of data being collected. Click on **Save** when you're ready.
2. Under **Type in a detailed description of the problem**, enter some text that provides information about the problem; then, reproduce the steps.
3. Next to **File Name:**, choose the path where you want to save the file to. By default, it will save to `%user%\Documents\WPR Files\`. Click **Save**.
4. Wait while the trace is being merged.
5. Click on **Open Folder**. There will be a file here, but you also want to include the folder.
6. Zip it up and upload it to the Microsoft CSS Support Engineer if you have a case open.

COLD SNACK

Try starting the trace at a whole minute – that is, **`/ 00`** *seconds (for example,* **`01:30:00`***, in the format of* **`HH:MM:SS`***). Look at a watch/clock so that you can write down when the issue reproduces. This will help when analyzing the data since if you are collecting this type of data, you are looking at a needle in a haystack.*

The next step is to analyze the **`.etl`** data. You can use the Windows Performance Analyzer (**`WPA.exe`**) for this, which provides a very useful user interface for diving deeper into the results of the trace. This is a great alternative to using PerfMon to capture data if you're not sure what to select.

Linux performance

There are a few key things to look out for on Linux machines when it comes to understanding performance. Many Linux machines have been sized to run the specific installed workload. Traditionally, many organizations do not run security solutions on Linux machines, and if they do, they are often very basic/lightweight. Examples include periodic malware scanning but no real-time protection, logging but no EDR, and so on.

Memory

First, let's talk about memory. Due to tight sizing, this is often one of the first areas where a machine gets into performance troubles. The way this presents itself is when the machine starts swapping – using the disk to write memory contents when there is no free memory to use.

The tricky part? It may *appear* that it's the CPU that is taking the brunt of performance, but it could very well be that significant disk swapping is the actual cause of the high CPU usage.

Run **`top`** to find out if the machine is using the swap file (at all) – if the usage of the swap file is above 2-3%, this is a strong indication the machine is under memory pressure and that before any further troubleshooting, you will need likely need to add more memory to rule out if the problem solely exists due to memory constraints.

Processor

Now that you've used the **`top`** command to rule out a memory problem, when it comes to CPU, you will need a little more background. For example, if you are observing **`wdavdaemon`** consuming 100% CPU usage, you may conclude that it is using up the entire CPU.

Run **`sudo cat /proc/cpuinfo`** to check how many cores the machine has first!

If you have an 8-core machine, 100% usage means one core is occupied – or ~13%. While this may be more than what you may expect to see (note that unless you have a baseline, you can only speculate), by itself, it doesn't mean there is a performance issue.

If you feel CPU consumption is still unexpectedly high, you can start ruling out if it's MDE causing it by temporarily disabling real-time protection, as follows:

```
sudo mdatp config real-time-protection --value disabled
```

If this alleviates the problem, this means that MDE real-time protection was responding to what was happening on the machine and exclusion can be considered – configure it before re-enabling real-time protection.

Audit

MDE on Linux uses the Linux Auditing Subsystem (and/or the **extended Berkeley packet filter** (**eBPF**)) – the **Audit Daemon** (**`Auditd`**) at **`/etc/audit/rules.d/`** contains the logging rules. Events added by MDE will be tagged with the **`mdatp`** key:

- High CPU resource consumption from the **`mdatp_audisp_plugin`** process is a good indicator that the EDR processes are having an impact
- **`/var/log/audit/audit.log`** increasing in size significantly or frequently rotating is another good indicator of this

Here, interactions between multiple security solutions are a common cause of performance issues. Otherwise, high transactional workloads can generate a flood of events. You will want to consider deduplicating security solutions or applying exclusions – you can place specific ones for **`Auditd`**. At the time of writing, MDE for Linux on newer kernel versions also supports the use of eBPF as an event source, and you may consider attempting to enable this to overcome performance issues with **`Auditd`**.

macOS performance

You can use **`top`** on macOS similarly to on Linux.

To narrow down whether Microsoft Defender Antivirus for Mac is contributing to the performance issues, you can consider temporarily disabling real-time protection.

If the device is not MDM managed, real-time protection can be disabled using one of the following options:

- **From the user interface**: Open the **Microsoft Defender** app and navigate to **Manage settings**
- **From the Terminal**: This requires elevation:

```
mdatp --config realTimeProtectionEnabled false
```

If the device is managed through MDM, real-time protection can only be disabled through the managed configuration file.

After disabling real-time protection and confirming it alleviates the issue, open **Finder** and click on **Applications** | **Utilities**. Open **Activity Monitor** and analyze which applications are using the resources on your system. Typical examples of apps with high resource consumption are software updaters and compilers.

Microsoft offers a download of a Python script to help you identify high CPU offenders. The output of this script will also allow you to zoom in on candidates for exclusions for the processes or disk locations that contribute to the performance issues.

COLD SNACK

Like on Windows, the client analyzer script offers the best tools to surface both AV and EDR-related issues and should be your go-to on both macOS and Linux.

Now that you've found the likely culprit, you may want to consider placing an exception. Read on to find out what options you have and their considerations.

Navigating exclusion types to resolve conflicting products

Once you find out where your impact is coming from, you may consider placing an exception. An exception impacts your security posture by creating blind spots, so you should always carefully consider if this is the best solution to address the issue at hand.

If the impact you are observing is performance/compatibility-related, a more permanent exception may be required.

COLD SNACK

Be mindful of interactions between security solutions. An(y) EDR is not a completely passive tool! EDR solutions use a variety of technologies, including hooking, that may trigger one tool to scan the other processes to see what's going on.

Running more than one solution that is inspecting what's happening on the machine is more than likely going to cause some impact that will be very hard to track down – let alone get support from either of the vendors. The same caveat applies to running more than one active antimalware solution – there's simply no way to accurately predict the outcome.

Don't get confused by passive mode – this only applies to running Defender Antivirus in a non-blocking way and is only intended to allow a different antimalware solution to provide antimalware capabilities while MDE's EDR is the active solution. Does that mean that running Microsoft Defender Antivirus in passive mode guarantees that there's no interaction with a non-Microsoft antimalware solution and MDE's EDR? **No***! You will likely have to exclude MDE's processes in that third-party antimalware solution.*

This next section will cover a few different ways you can narrow down which exception is needed – indicators or exclusions.

Submitting a false positive

When you encounter a **false positive** (**FP**), meaning a legitimate file or process in your organization got flagged as malware and even quarantined by MDE, you will want to report this to Microsoft as soon as possible.

You can submit an FP directly to Microsoft from the `security.microsoft.com` portal, or through the **Microsoft Malware Protection Center** (**MMPC**).

Exclusions versus indicators

There are various tools at your disposal to mitigate issues. Assuming you are not facing an FP (see the previous section) and you need to place an exception, you can consider using either indicators or exclusions. The difference lies in how a file is evaluated or scanned, by which component, and what the output will be in your portal.

Here are some common reasons for considering exceptions that are not FPs:

- CPU spikes in `MsMpEng.exe`
- Application startup delays
- Application takes longer to do the same work
- You are running other security solutions that interact (negatively) with MDE

When would you select one over the other? This depends on the symptom. If you've used the MDAV performance analyzer and/or PerfMon and narrowed down the component involved, you should already know where to place an exception (or if you need a support case).

If you haven't narrowed it down yet, here are some high-level strategies for an alternative troubleshooting approach:

- Temporarily turn off real-time protection. If this alleviates the problem, you know that MDAV is in play. Similarly, if the problem only occurs during scheduled (full/quick) scans, you know where to start; you're likely headed toward an AV exclusion.
- If turning off real-time protection doesn't help, and you're observing other MDE processes being very active, this would point to an interac-

tion with EDR. Check if there are any other security solutions active and turn them off before troubleshooting further.

Here are the different possible exceptions and what you would use them for:

- **Alert suppression**: This should be used if you don't want to create a blind spot but you also don't want to generate any alerts. This is typically used for FPs (you should still submit to Microsoft). This is preferred over **allow** indicators!
- **AV exclusions**: This is a workaround for mostly performance or compatibility-related challenges. Know that this will create a blind spot in your protection. Always make these as specific as possible and reassess them periodically.
- **Allow indicators**: This option will allow a file to run, but MDAV will still scan it, so this may not solve performance-related issues. You can allow by hash or by signer (certificate). Know that this will create a blind spot in detection.

Contextual exclusions

If you've eliminated an FP and decided to place an exclusion for MDAV, try to be as specific as possible. Contextual exclusions are a great way to be more deliberate. Let's cover a few tips for handling those:

- Determine if you intend to exclude a **file path** or a **folder path**. The difference between a file and a folder may be obvious to you, but when you are specifying an exclusion, MDAV does not make a distinction between them. This means that, as an attacker, I could create a file with the same name as an excluded folder or vice versa to escape detection.
- Similarly, determine if what you need is a process exclusion instead of a path exclusion. A file/folder exclusion means that the excluded path (file/folder) will not be scanned (for example, `c:\folder\program.exe`); a process exclusion (files opened by a

process) means files that are *touched by* the excluded process (**`c:\folder\program2.exe`** is opening files in **`c:\folder\files`**) will not be scanned.

COLD SNACK

Be very careful with process exclusions as they can affect a large number of files – for example, if you exclude **`mydatabase.exe`***, anyone can spawn a process by that same name and it would be allowed to perform any file operations anywhere on the filesystem without them being scanned. At the very least, attempt to provide the full path to the process or, even better, match the process (with the full path to it!) to the files it should be expected to use.*

Because of this, process exclusions only apply to real-time protection and not to scheduled or on-demand scans (quick/full/resource).

A very common performance-related scenario is where a specific process is performing a large volume of writes to a file – database software is probably the most evident example of this. What you will want to avoid is excluding the process (see the preceding *Cold snack* note) entirely – instead, you can specify both the process and its full path, as well as the files or folder together, and you can even limit the exclusion so that it only applies to, for example, real-time protection.

Linux and macOS

For both Linux and macOS, you can use the command line or a managed configuration to place AV exclusions. The XMDEClientAnalyzer tool is likely the best way to identify what exactly to exclude.

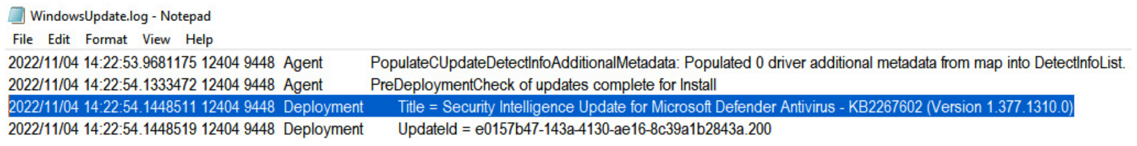
On Linux, if disabling real-time protection does not alleviate the issue, the XMDEClientAnalyzer tool will help you understand if you need to place **`Auditd`** exclusions. These exclusions only apply to the EDR sensor.

Let’s look at the different ways we can check how machines are bringing down updates and from where.

Understanding your update sources

When you don’t want, for example, your Windows servers getting Defender security intelligence updates directly from Windows Update, because you want to be more in control, here’s how you can find out where updates are coming from:

1. Check the logs. If you see an entry in any of the logs collected by `mpcdmrun -getfiles` (`MpCmdRun.log`, `MpCmdRun-LocalService.log`, `MpCmdRun-NetworkService.log`, and `MpCmdRun-System.log`), you know that updates are coming from ConfigMgr **Windows Server Update Services** (**WSUS**), WSUS standalone, the MMPC, or a file share.
2. If you’ve configured a policy to check for security intelligence updates at a specific interval, check the Windows Defender Operational event log. If you see that a *security intelligence update* is being applied, while none of the logs from *step 1* show as downloading the file, you can probably conclude the update is coming from Microsoft Update directly.
3. To confirm, open `WindowsUpdate.log` in your Windows folder or by running `Get-WindowsUpdateLog` using PowerShell. You will find entries like the following:



```
2022/11/04 14:22:53.9681175 12404 9448  Agent         PopulateCUpdateDetectInfoAdditionalMetadata: Populated 0 driver additional metadata from map into DetectInfoList.
2022/11/04 14:22:54.1333472 12404 9448  Agent         PreDeploymentCheck of updates complete for Install
2022/11/04 14:22:54.1448511 12404 9448  Deployment      Title = Security Intelligence Update for Microsoft Defender Antivirus - KB2267602 (Version 1.377.1310.0)
2022/11/04 14:22:54.1448519 12404 9448  Deployment      UpdateId = e0157b47-143a-4130-ae16-8c39a1b2843a.200
```

Figure 9.8 – Snippet from the WindowsUpdate.log file

COLD SNACK

You should consider building a green room to try reproducing the issue outside of your domain or Azure AD environment. Even if you can't reproduce the issue, you can start eliminating possibilities and generate logs for a working state, to compare. One way of getting a representative green room is to grab an image of a working machine – check out the Disk2VHD tool at **https://learn.microsoft.com/en-us/sysinternals/downloads/disk2vhd** *for more.*

Next, let's talk about comparing files as a form of troubleshooting. By doing this, we can understand what could have changed to introduce an issue.

Comparing files

As an alternative to debugging, comparing files is a great way to understand what has changed – to zoom in on what is causing a problem. This is particularly useful when you are trying to reproduce an issue and can't reproduce it on a different machine: if you have a log of the issue when it first occurred, and you have a log of an attempted reproduction, you can consider the failed reproduction as the normal state (working as intended) and play spot the difference.

Some popular tools for file comparison are FC.exe, WinDiff.exe, Visual Studio Code, Notepad++, and BeyondCompare.

How do you go about this? First, make sure you understand which log is of particular interest. This may be one of the log files that gets captured using the `mpcmdrun.exe -getfiles` command – for example, `MPRegistry.log` in `C:\ProgramData\Microsoft\Windows Defender\Support`.

Like with any reproduction, you will want to get to a *clean* state first, delete the log file, reproduce the issue, and then perform the same steps on a different machine. If it doesn't reproduce there, great! You now have two logs to compare – one containing the failure and one containing the normal (expected) state.

Bonus – troubleshooting book recommendations

Much of this chapter materialized because of input from Yong Rhee, a Microsoft employee that has built a reputation for being an elite troubleshooting guru, back from his days as a **Premier Field Engineer** (**PFE**) (now called a **Customer Engineer** (**CE**).

The following are recommendations for books that Yong considers essential reading if you want to further develop your troubleshooting skills:

- *Troubleshooting with the Windows Sysinternals Tools, 2nd Edition*:

https://www.microsoftpressstore.com/store/troubleshooting-with-the-windows-sysinternals-tools-9780735684447

- *Windows Performance Analysis Field Guide*:

https://www.elsevier.com/books/windows-performance-analysis-field-guide/huffman/978-0-12-416701-8

- *Windows Internals Book*:

https://learn.microsoft.com/en-us/sysinternals/resources/windows-internals

- *Network Monitoring and Analysis: A Protocol Approach to Troubleshooting*

With some recommendations in mind, let's close out this chapter by summarizing what was covered.

Summary

In this chapter, you learned how to troubleshoot common issues you may encounter when setting up or running MDE by covering areas such as connectivity and ways to use the client analyzer, as well as traces. We covered onboarding indicators, which help us understand failures, policy precedence, and all sorts of performance troubleshooting.

All security solutions come with a trade-off: you sacrifice (some) performance for security. If the balance shifts and you need to investigate, knowing how to troubleshoot will help you quickly identify where the problem is coming from. To ensure that security is always maintained, it pays to find and solve the cause as opposed to simply disabling capabilities.

In the next chapter, you will find more useful background information that can help you even better understand the internals of the product or, if you are familiar with the product, find out useful tips and tricks to further optimize your *operational excellence*.