packt

1ST EDITION

Microsoft Defender for Endpoint in Depth

Take any organization's endpoint security to the next level

PAUL HUIJBREGTS | JOE ANICH | JUSTEN GRAVES

Microsoft Defender for Endpoint in Depth

Take any organization's endpoint security to the next level

Paul Huijbregts

Joe Anich

Justen Graves

‹packt›

BIRMINGHAM—MUMBAI

Microsoft Defender for Endpoint in Depth

Group Product Manager: Mohd Riyan Khan
Publishing Product Manager: Mohd Riyan Khan
Senior Editor: Romy Dias
Technical Editor: Shruthi Shetty
Copy Editor: Safis Editing
Project Coordinator: Ashwin Kharwa
Proofreader: Safis Editing
Indexer: Hemangini Bari
Production Designer: Prashant Ghare
Marketing Coordinator: Ankita Bhonsle

First published: January 2023

Production reference: 1020223

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham
B3 2PB, UK.

ISBN 978-1-80461-546-1

`www.packtpub.com`

Contributors

About the authors

With almost 20 years of industry experience and relevant certifications, **Paul Huijbregts** has a long history of working with customers across the world leveraging his passion for (Microsoft) security solutions – and being brutally honest about them.

After joining Microsoft in 2016 and engaging regularly with Defender for Endpoint teams, Paul moved to Redmond (together with his wife and kids) to join them and become a product manager – in the middle of the pandemic (October 2020). Here, he is on what is called the "Platforms" team, working on solutions across operating systems and environments, focusing primarily on server endpoints and security management. His motto is: "I drink beer and I know Microsoft security things."

I would like to thank my wife and kids for giving me the space and time (and the beer money) required to keep writing. In addition, big thanks to the infosec community for their continued support, my peers, and most of all some of the excellent subject matter experts that have been working with the product for much, much longer than I have – thanks for your passion, dedication, and entertaining this crazy PM that was asking lots of questions over lunch or coffee.

Joe Anich has 15 years of experience in the IT industry ranging from endpoint management with a focus on SCCM and Intune to endpoint security and incident response. Currently working on Microsoft's **Detection and Response Team** (**DART**), he works closely with customers during critical moments. Working in incident response has given Joe insight into SOC operations and how to help teams around the world improve their security posture as a whole. Outside of work, Joe enjoys running around the house with his 2-year-old son playing "chase me." Fun fact: During the late 90s, Joe could be found at the roller-skating rink most Friday nights, gliding around the rink with a super rope in hand, maybe in JNCOs or Lee Pipes, vibing to 90s hip hop.

I want to thank my beautiful wife, Katie, for running the household during my chaotic work schedule and yet still allowing me to pursue my passion to write this book in whatever hours were left of the day. My success comes from your willingness to support me. Thanks for all you do, and for being the best mother little Z could ask for.

Justen Graves is a security engineer with 14 years of IT experience. Most of his career has been focused on endpoint enablement and security, with the last 4 years spent at Microsoft. Currently working in Microsoft's Cyber Defense Operations Center, their internal SOC, he uses tools such as Microsoft Defender for Endpoint every day to defend corporate Microsoft from attack.

Justen has a BS in cybersecurity and an MBA. He holds many industry certifications, including CISSP, PMP, and GSEC, and several Microsoft certifications, including Azure Solutions Architect Expert and Enterprise Administrator Expert. Starting his career at Walmart and managing to never relocate, he resides in Northwest Arkansas with his wife and three children.

I want to thank my beautiful wife, Paula, for all the support and compassion while I struggled through this book... immediately after an MBA. She and my children, Andrew, Sloane, and Ember, as well as my mom, Sharon, were incredibly supportive and patient with me throughout this time and I truly couldn't have done it without them.

I'd also like to thank the DSR SOC team and leadership for their support, as well as my Microsoft family for all the knowledge shared with me along the way. It would take pages to list all the fantastic people I've encountered at Microsoft. I do want to show explicit appreciation to Joe and Paul for letting me come along on this journey, without whose perseverance this book never would have happened.

3
Introduction to Attack Surface Reduction

In this chapter, we will address which additional layers of defense can be applied to your endpoints for additional opportunities to prevent attacks from gaining a foothold. Elements of this layered defense include the prevention of certain user- or application-initiated actions but also blocking connections to bad destinations, including those in use by attackers that have already had some success gaining a level of control over a device. Since some of these additional controls can have an impact on the user experience, your business software, or other (security-related) tools, you may need to carefully consider which can be safely applied.

We will cover the following topics:

- What is **attack surface reduction** (**ASR**)?
- Examining **ASR rules**
- **Network protection** (**NP**) layers and controls
- **Controlled folder access** (**CFA**) ransomware mitigations
- Exploit protection for advanced mitigations

Cold snack

ASR features used to fall under the category name **exploit guard**, and you can still see that in various places today, including in **Microsoft Configuration Manager** (**ConfigMgr**).

What is attack surface reduction?

The attack surface for an organization is comprised of all possible attack vectors that can be used by attackers to gain unauthorized access to the assets or data owned by that organization. It usually involves exploiting vulnerabilities or abusing weaknesses and loopholes in the places and resources owned or accessed by the organization.

ASR is a process of minimizing the ways in which attackers can perform successful intrusion attacks on protected assets. In fact, preventing assets from getting compromised and minimizing the extent of the impact in the case of a successful compromise are the two primary responsibilities of ASR. The mitigations used as part of this process sometimes lead to limited acceptable impact on the capacity and usability of the assets being protected. Nevertheless, ASR remains one of the most fundamental requirements for maintaining a strong security posture and is often seen as the first line of defense against many attacks.

In MDE, many ASR capabilities apply to user endpoints; you will find that most ASR rules, for example, apply to user software/interactions, and are used to prevent activities that could lead to compromise. This is also an area where you can run into disruption to productivity, just because some legitimate business software behaves a little differently. But they can also target certain operations, typically performed by users, that should not occur often—if at all—on machines managed by an organization.

These capabilities can be best described as **host intrusion prevention systems** (**HIPS**)-like if you wish to compare them against prevalent endpoint protection solutions. However, it is important to note that the intent of these capabilities is quite different from traditional HIPS in that they are intended to, based on extensive research, strike the best balance between security and impact on, for example, performance or productivity. They are not intended to provide a framework for creating your own rules; instead, these are very targeted measures and, in most cases, should go unnoticed by the end user.

Conversely, some of the capabilities in this category act on a system level, providing overall protections such as against malicious outbound connections, access to credentials, executables from unverified sources, vulnerable drivers, and disk-level modifications—typically considered to be undesirable avenues worthy of placing restrictions on in a managed environment. These are worth universally applying to any type of endpoint.

Microsoft Defender for Endpoint (**MDE**) is a complete suite that contains many security features; ASR is one of the category names that essentially applies to any feature that sits outside of the core premise of next-generation protection (which was covered in the previous chapter).

Examining ASR rules

We will talk about the general philosophy behind ASR rules, followed by each of the rules, and how we group them.

The philosophy behind ASR rules

In general, from a usability point of view, the **HIPS** is expected to fulfill two key objectives. Primarily, it should allow the creation of behavior-based rules for blocking specific activities, and it should provide a way to mitigate or manage the adverse impact in case of a **false positive** (**FP**). Exclusions are usually the most common method used for the same. So, the aim is to provide organizations with the control and flexibility required for managing the security posture of protected assets. This makes HIPS the obvious choice for ASR. Therefore, as a common industry practice, **security providers** (**SPs**) have been building platforms that allow the creation and management of rules and exclusions, and organizations are often seen making use of them throughout their tenures with their providers.

However, there are several downsides to this approach. It has been observed that a lot of times, a lot of these rules are created in the nick of time in response to first-party security incidences or third-party attack investigations and publications. As such, many of these rules end up being defined using **indicators of compromise** (**IoCs**) uncovered during the studied attacks. Quite often, this makes them too specific and, in the face of continuously evolving attack kill chains, incapacitated in their ability to block successive variants of the underlying attacks. This leads to the creation of the next batch of the rules (or updates to the existing rules), and the cycle continues only to stop and start again when the focus shifts to a new incidence, blog, tweet, or some security news. As a result, the rules and the updates keep piling up, and soon the organizations find themselves dealing with hundreds or even thousands of such rules but still struggle with fending off new attacks.

Some of the most common challenges faced when working with a HIPS product include the following:

- Researching and authoring generic rules that could proactively block previously unseen attacks
- Managing the adverse impact using exclusions on first-party or third-party **line-of-business** (**LOB**) apps caused by the FPs of generic rules
- Retiring **end-of-life** (**EOL**) rules in a timely manner

Even when available time is not an issue, the lack of visibility beyond the boundaries of your own organization becomes the limiting factor for authoring generic proactive rules with meaningful exclusions in a timely manner.

Microsoft has used a slightly different approach while designing its take on the ASR feature. MDE provides a set of predefined rules that organizations can deploy and manage on their own. Microsoft recognizes that, by virtue of billions of signals received from millions of devices on daily basis, they are in a unique position to determine the extent of the impact of a certain blocking rule. As such, instead of only building the platform and completely offloading the responsibility of creating rules onto their customers, Microsoft partners with them and shares that responsibility.

However, as stated previously, preventing assets from getting compromised and minimizing the extent of the impact in case of a successful compromise are the two primary responsibilities of ASR. Therefore, to come up with a set of meaningful rules that are generic enough to block an entire class of attacks, the researchers at Microsoft categorized attacks based on initial point-of-entry vectors and defined one generic rule for each group. This ensured that not only were the known attacks blocked but the ones that were not previously seen and used the same underlying point-of-entry vectors were also blocked successfully. For example, many attacks were seen originating from Microsoft Office (Office) files, specifically from the **Visual Basic for Applications** (**VBA**) macro code that made use of Win32 API calls to perform malicious activities. The **Block Win32 API calls from Office macros** rule was created to block the execution of all such macros.

The same approach has been followed for minimizing the extent of compromise post-intrusion. For example, it was observed that in the case of many attacks, after the device had been compromised, to further compromise the network, the attackers tried to steal credentials from `LSASS.exe` process memory. The **Block credential stealing from the Windows local security authority subsystem (lsass.exe)** rule was created to prevent all processes from accessing the **local security authority subsystem service** (**LSASS**) process memory, thus limiting the spread of all such attacks that relied on stealing credentials from the LSASS process.

Rule categories and descriptions

Microsoft has developed several ASR rules and continues to evaluate the possibilities of creating more, to help MDE customers manage their attack surfaces. Here is the list of the existing rules and their GUIDs from `https://docs.microsoft.com/en-us/microsoft-365/security/defender-endpoint/?view=o365-worldwide` for ASR:

Rule Name	**Rule GUID**
Block abuse of exploited vulnerable signed drivers	`56a863a9-875e-4185-98a7-b882c64b5ce5`
Block Adobe Reader from creating child processes	`7674ba52-37eb-4a4f-a9a1-f0f9a1619a2c`
Block all Office applications from creating child processes	`d4f940ab-401b-4efc-aadc-ad5f3c50688a`
Block credential stealing from the Windows local security authority subsystem (lsass.exe)	`9e6c4e1f-7d60-472f-ba1a-a39ef669e4b2`
Block executable content from email client and webmail	`be9ba2d9-53ea-4cdc-84e5-9b1eeee46550`
Block executable files from running unless they meet a prevalence, age, or trusted list criterion	`01443614-cd74-433a-b99e-2ecdc07bfc25`
Block execution of potentially obfuscated scripts	`5beb7efe-fd9a-4556-801d-275e5ffc04cc`
Block JavaScript or VBScript from launching downloaded executable content	`d3e037e1-3eb8-44c8-a917-57927947596d`
Block Office applications from creating executable content	`3b576869-a4ec-4529-8536-b80a7769e899`
Block Office applications from injecting code into other processes	`75668c1f-73b5-4cf0-bb93-3ecf5cb7cc84`
Block Office communication application from creating child processes	`26190899-1602-49e8-8b27-eb1d0a1ce869`
Block persistence through Windows management instrumentation (WMI) event subscription	`e6db77e5-3df2-4cf1-b95a-636979351e5b`
Block process creations originating from PsExec and WMI commands	`d1e49aac-8f56-4280-b9ba-993a6d77406c`
Block untrusted and unsigned processes that run from USB	`b2b3f03d-6a65-4f7b-a9c7-1c7ef74a9ba4`
Block Win32 API calls from Office macros	`92e97fa1-2edf-4476-bdd6-9dd0b4dddc7b`
Use advanced protection against ransomware	`c1db55ab-c21a-4637-bb3f-a12568109d35`

Table 3.1 – ASR rules with associated GUIDs

Depending on the area of operations, the ASR rules could be categorized into groups that make it easier if you are looking to harden any area first.

Productivity app rules

These rules are used to block behaviors or attacks that attempt to exploit productivity applications:

- Block Office applications from creating executable content
- Block Office applications from creating child processes
- Block Office applications from injecting code into other processes
- Block Win32 API calls from Office macros
- Block Adobe Reader from creating child processes

Block Office applications from creating executable content

In attacks that involve exploiting vulnerabilities in Office processes or abusing the features and functionalities of Office applications, one of the common successive stages is to drop and execute a malicious file on the affected device. This is where the attacker succeeds in successfully intruding into the device and can perform any number of malicious activities from this point onward.

This rule prevents Office applications, Word, Excel, PowerPoint, and OneNote from dropping the executable content on the disk. In doing so, the rule aims to block the attacks at a crucial stage where attackers are looking to gain access and obtain a foothold on the machines.

In the case of Windows executables files—that is, **Portable Executable** (**PE**) files, the rule only blocks untrusted and unsigned files from being written to the disk. This prevents some of the intended use cases from being blocked. However, the script files are outright blocked without validating trust or friendliness status.

A few popular applications have already been excluded from the rule using backend exclusions or **global exclusions**. However, to protect you from attacks that may abuse these exclusions, Microsoft has intentionally refrained from publishing a list of the excluded applications/processes. Despite deploying global exclusions, you may still need to deploy local exclusions for first-party internal or third-party LOB applications blocked by the rule.

Block all Office applications from creating child processes

As mentioned earlier, attackers that use Office applications—that is, Word, Excel, PowerPoint, OneNote, and Access—as the point-of-entry vector often attempt to download and execute malicious files on the affected device, or Office processes are used to execute system processes or admin tools or benign third-party processes to deepen or broaden the infection—for example, launching Command Prompt or PowerShell to disable an important security control or make some registry changes.

As such, this rule provides another important control—it blocks all Office applications from launching any child processes. No trusted files, friendly files, system files, admin tools, or benign third-party applications are allowed to execute.

Block Office applications from injecting code into other processes

Injecting code into legitimate clean processes (mostly, signed processes) is a well-known detection evasion technique. Office processes have not been immune to this technique. However, the researchers at Microsoft understand that there is no good reason for Office processes to inject code into other running processes. This rule blocks processes related to Word, Excel, and PowerPoint Office applications from performing code injection activity.

As with the rest of the productivity app rules, this rule also supports ASR exclusions.

Block Win32 API calls from Office macros

The Office VBA macro is an extremely popular feature among Office users. However, even though Office supports calling Win32 APIs from inside the VBA macro code, most organizations do not make use of the functionality as often. On the other hand, attackers can use it to perform a host of malicious activities, including conducting a fileless execution of the malicious shell code, making it exceedingly difficult to detect. This rule blocks the execution of macro code that contains Win32 API calls.

The rule supports exclusions. As such, the organizations that need macros to make Win32 API calls can exclude specific Office files and continue to block the rest.

Block Adobe Reader from creating child processes

As with Office applications, attackers can also use various techniques to perform attacks using Adobe Reader processes. In such attacks, the Adobe Reader process often launches additional payloads or admin tools. This rule blocks such attacks by blocking Adobe Reader from launching child processes.

As with Office rules, a few popular applications have already been excluded from the rule using global exclusions. As always, you can also deploy on-client exclusions.

Script rules

These two rules are focused on mail clients creating and launching executable content, something very popular with malicious documents sent in phishing or spearphishing campaigns:

- Block execution of potentially obfuscated scripts
- Block JavaScript/VBScript from launching downloaded executable content

Block execution of potentially obfuscated scripts

Obfuscating scripts (for example, containing JavaScript/VBScript/**PowerShell** (**PS**)/macro code) is a customary practice among script authors. It provides a way to obscure the contents for protecting proprietary information, as sometimes it may include intellectual property. However, malware authors abuse this capability to make their malicious scripts difficult to read, analyze, and detect.

This rule looks for properties that are indicative of obfuscation and uses **machine learning** (**ML**) models on top of the identified properties to classify and successively block the scripts.

Since the rule makes use of ML models, there is always a certain degree of FPs associated with this rule. To mitigate the adverse impact of such FPs, such as productivity app rules, this rule also supports exclusions.

Block JavaScript/VBScript from launching downloaded executable content

Downloading malicious scripts or binaries and using them to perform malicious activities during the stages of an attack is quite a widespread practice. As the name suggests, the aim of this rule is to block JavaScript and VBScript scripts from launching downloaded malicious content. Once enabled the rule blocks such scripts from executing on the protected device.

However, several genuine scenarios need scripts to be able to download and execute hosted content. As such, the rule also supports ASR exclusions.

Communication app rules

These rules are intended to reduce the chance that communication applications are abused as an initial attack vector:

- Block executable content from email client and webmail
- Block Office communication applications from creating child processes

Block executable content from email client and webmail

In attacks that involve email as the vector, it is often seen that some sort of malicious executable file is delivered to the affected device in the form of an email attachment. Therefore, this rule has been created to block all executable files (such as `.exe`, `.dll`, and `.scr`) as well as popular script files (such as `.ps1`, `.vbs`, and `.js`) delivered via email. When such emails are opened and the executable or script file is accessed, the rule inspects and blocks the files in question.

The rule works for the Outlook client application as well as Outlook.com and other popular webmail providers.

In general, most organizations do not approve of sharing executable or script files over email. However, there is always a pocket of users or devices where the activity is observed and successively either allowed or simply ignored. It is never recommended to allow such sharing over email; however, in case you have a need to allow this, at least for a small set of devices, this rule also supports ASR exclusions.

Block Office communication applications from creating child processes

As mentioned earlier, email has been used as a point-of-entry vector for a long time. Blocking the executable files delivered using email is just one scenario. Other popular scenarios include exploiting vulnerabilities or abusing loopholes in the Outlook app. In such cases, the attack kill chain usually contains the Outlook app launching some processes on the affected devices. In most cases, these are admin tools that can get abused to lower the security posture of the device and download additional malicious binaries or tools needed for the attack.

This rule protects against such attacks by blocking all child processes created by the Outlook app.

Note that as with the Office-related rules from the *Productivity app rules* section, a few popular applications have already been excluded from the rule using global exclusions. The rule also supports ASR exclusions. Users can always exclude internal or LOB apps blocked by this rule.

Polymorphic threat rules

These rules focus on preventing new files from running:

- Block executable files from running unless they meet a prevalence, age, or trusted list criteria
- Block untrusted and unsigned processes that run from USB

Block executable files from running unless they meet a prevalence, age, or trusted list criteria

The one fact unanimously agreed upon and repeated multiple times by many SPs is that most Windows executable files used by attackers in conducting attacks are polymorphic in nature. So, it is almost certain that the binary files used in any attack are going to be new files. Their file hashes would have never been recorded by SPs in the past. This polymorphism is often a result of packing and obfuscating the binary files—techniques used to obscure contents and make the files difficult to reverse engineer, analyze, and detect using signatures. The technique also makes it easier to evade detections.

Microsoft researchers combined the fact that these files are always new and non-prevalent with the fact that they are also never valid-signed by a genuine entity, and this ASR rule was born. The rule blocks all new and non-prevalent files that are not already trusted or valid-signed. To verify the trust, the rule consults Microsoft's reputation system. As such, cloud-delivered protection must be enabled so that the rule can function as expected.

This is an aggressive rule that blocks new files if they are untrusted or unsigned, at least until they have gained prevalence or trust in the Microsoft customer base. As such, users are always advised to deploy the rule on devices where this level of aggressive blocking is desired (high-value assets with limited new app or update installations) or on devices where there is some room to absorb the impact of some FPs. Devices on which new and non-signed binaries are expected to arrive on a frequent basis, such as build systems and developer machines, should only be included after excluding the folders related to such files/activities.

Block untrusted and unsigned processes that run from USB

There are many examples of attacks conducted by delivering malware using a USB drive. This rule prevents such attacks by blocking all unsigned and untrusted executable files from executing from the USB drive. The rule also tracks executable files that have been copied on the disk from the USB drives and blocks them from executing.

The rule also supports ASR exclusions. This allows users to exclude files that are intended to be distributed within the organization using USB drives.

Human-operated ransomware rules

This category covers rules that are directed against ransomware:

- Block abuse of exploited vulnerable signed drivers
- Use advanced protection against ransomware

Block abuse of exploited vulnerable signed drivers

This is one of the most recommended ASR rules for protecting against ransomware attacks.

There have been several incidences where attackers used signed vulnerable drivers for gaining kernel-level access. The drivers were dropped by the attackers on the devices in question, to conduct these attacks.

This rule prevents vulnerable drivers written to the disk from being loaded by any process, thus making them inaccessible for all processes running on the system.

Cold snack

The rule does not block *in-use* copies of vulnerable drivers. It only blocks copies newly written to the disk. This helps to avoid app crashes or blue screens. Even though users are never really expected to make use of them, the rule also supports ASR exclusions.

Use advanced protection against ransomware

This is an aggressive ASR rule for protecting against ransomware, or to quote `https://docs.microsoft.com/en-us/microsoft-365/security/defender-endpoint/attack-surface-reduction-rules-reference`, the rule tends to err on the side of caution to prevent ransomware.

The rule looks for files with properties that are most commonly found in ransomware files and blocks the execution of such files if they are not valid-signed, not prevalent across the Microsoft customer base, not very old—that is, the first seen date is relatively recent—and not considered as a clean or a friendly file by the Microsoft Defender cloud-based file reputation system. As such, it is evident that the rule needs cloud-delivered protection enabled.

Just as with the **Block executable files from running unless they meet a prevalence, age, or trusted list criterion** rule, this is an aggressive rule that uses heuristic detection approaches to identify the files to be blocked. Therefore, it is expected to have a certain degree of FPs. However, just as with the other rules, exclusions are supported.

Lateral movement and credential theft rules

In this category, rules are responsible for countering lateral movement techniques and credential abuse:

- Block process creations originating from PsExec and WMI commands
- Block credential stealing from the Windows LSASS (`lsass.exe`)
- Block persistence through WMI event subscription

Block process creations originating from PsExec and WMI commands

PsExec and WMI allow remote code execution. This has been abused by attackers in several attacks for spreading within networks in the affected organizations.

To fulfill the promise of minimizing the extent of the impact, post-compromise, it was essential to build a rule to prevent the lateral movement of attackers within the networks using either the PsExec or WMI functionalities. This rule provides that ability by blocking PsExec and WMI from launching child processes.

Cold snack

This rule will block the WMI commands that ConfigMgr uses to manage devices. You should not use this rule if you are managing those systems with ConfigMgr in any way, including co-management (using Microsoft Intune and ConfigMgr in tandem). The rule supports ASR exclusions. See `https://learn.microsoft.com/en-us/intune`.

Block credential stealing from the Windows LSASS (lsass.exe)

One of the essential steps for moving laterally within the network is to steal the credentials required for logging in on the target devices. Attackers have been seen targeting the Windows LSASS (`lsass.exe`) and stealing credentials from the `lsass.exe` process memory.

This rule prevents all processes from accessing the process memory of `lsass.exe`, thus making it impossible to steal credentials from the same. This makes it another very good mitigation against attacks, and attackers looking to move laterally within the org.

This rule only blocks processes from accessing the memory of `lsass.exe`—it does not affect any other functionality of the blocked processes. As such, excluding blocked applications is not required and therefore is not recommended. Here is a note in that regard: `https://docs.microsoft.com/en-us/microsoft-365/security/defender-endpoint/attack-surface-reduction-rules-reference`.

Cold snack

This rule can generate a lot of noise, but that doesn't mean there's an active threat. Some applications have simply been written to enumerate all running processes and try to open them with exhaustive permissions. Try to avoid adding an exclusion if there's no impact on the operation of the application to preserve visibility on true malicious operations.

An interesting comment in the note is that *this rule can generate a lot of noise*. This is mainly due to the noted fact—process enumeration activity is performed by many clean applications. As such, to handle the high volume of events generated by the rule, Microsoft has deployed the event deduplication logic specifically for this ASR rule. By virtue of the same, the events are de-duped within a window of a certain number of hours (the exact number is not publicly disclosed). For users, it means that for every block from the rule that is seen in the **Protection History** tab in the Windows Security app or seen in the telemetry queried using **Advanced Hunting** (**AH**) (more details in the *Analyzing ASR telemetry using AH* section), or in the **Event Tracing for Windows** (**ETW**) events accessed using Event Viewer, there could have been many more occurrences of the same block, within the next few hours of the reported event, that have simply not been reported for reporting and telemetry optimization purposes.

Cold snack

Since the functionality of the clean apps blocked by the rule is not adversely impacted in any way, and therefore since you are not expected to deploy any exclusions (unless you just don't want to see the apps being blocked), this rule has also been deployed in the on-by-default state by Microsoft. Again, it only means that the rule is auto-enabled in **block** mode on all devices where the rule hasn't been explicitly configured. However, for devices on which the default mode is changed to something else, such as **warn**, **audit**, or **disabled**, the decision is honored, and the rule is never switched automatically to **block** mode. Even though they are not recommended, the rule supports ASR exclusions.

Block persistence through WMI event subscription

This rule prevents attackers from abusing the WMI repository and events for deploying persisted malicious code. Since the blocked items are not really the files on the disk, the rule does not support ASR file and folder path exclusions.

Operating modes

Pursuing a strategy to reduce the attack surface involves understanding what this surface is in the first place. That is often easier said than done—and why HIPS solutions are hard to get right. Luckily, MDE does help in several ways to not just identify the attack surface (see *Chapter 7*, *Managing and Maintaining the Security Posture*); it also allows you to take a step-by-step approach toward implementation by providing **audit**/**warn** modes. These modes will allow you to observe the potential impact on your users/applications before you decide to roll out broadly, with exceptions in place where required.

ASR originally supported three modes—namely, **disabled, audit**, and **block** modes. However, a new mode called the **warn** mode was also added to provide the break-the-glass scenario.

Disabled mode

At any given point in time, an ASR rule can be found in one of the two states: **configured** and **not configured**. In the **not configured** state, the rule performs no action.

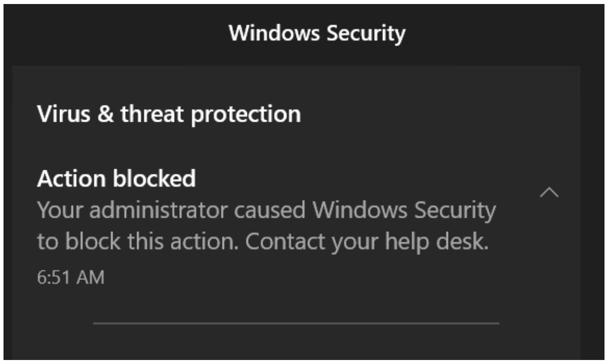
When set to **disabled** mode, the rule still does not perform any action; however, since it has been explicitly configured to take no action, the state of the rule changes from **not configured** to **configured**.

Audit mode

This is ASR's evaluation mode. In this mode, the rule does not block the targeted activity. It only generates an ASR audit ETW event every time the targeted activity is observed. This mode is useful for evaluating the impact of the rule and deploying appropriate mitigations if required, before enabling the rules in **block** mode.

Block mode

In this mode, the rule is enabled to block the targeted activity. A pop-up toast notification is generated every time the rule blocks some activity. The block event also gets added to **Protection history** under the Windows Security app's **Virus & threat protection** tab. An ASR Block ETW event is generated and logged, as shown in *Figure 3.1*:

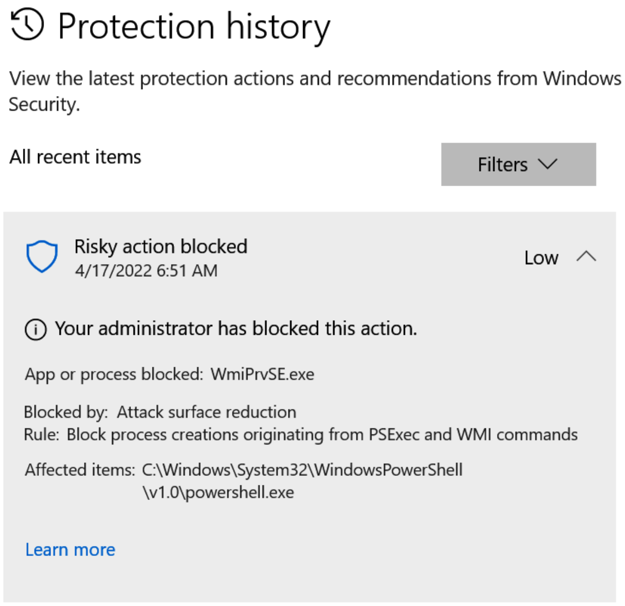

Windows Security

Virus & threat protection

Action blocked
Your administrator caused Windows Security to block this action. Contact your help desk.
6:51 AM

Figure 3.1 – Block mode notification

In *Figure 3.2*, we can see the same action that was blocked in the Windows Security notification but in the Windows Security app:


Protection history

View the latest protection actions and recommendations from Windows Security.

All recent items

Risky action blocked
4/17/2022 6:51 AM

Low

Your administrator has blocked this action.

App or process blocked: WmiPrvSE.exe

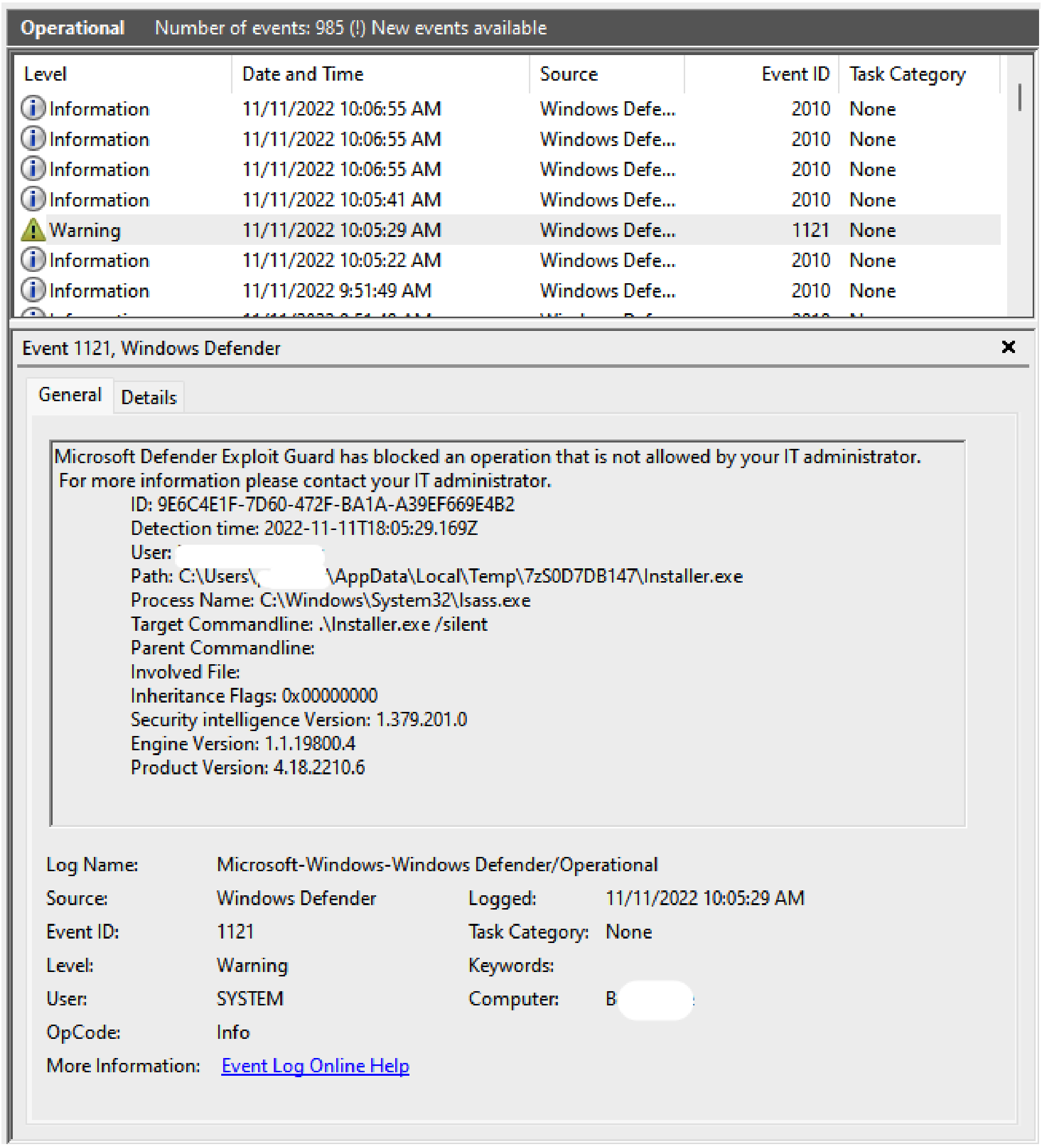
Blocked by: Attack surface reduction
Rule: Block process creations originating from PSExec and WMI commands

Affected items: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

Learn more

Figure 3.2 – ASR blocked item enlisted under the Windows Security app's Protection history

And for further information on blocks, you can view **Event Viewer** for logged entries, as shown in the following screenshot:



Operational Number of events: 985 (!) New events available

Level	Date and Time	Source	Event ID	Task Category
Information	11/11/2022 10:06:55 AM	Windows Defe...	2010	None
Information	11/11/2022 10:06:55 AM	Windows Defe...	2010	None
Information	11/11/2022 10:06:55 AM	Windows Defe...	2010	None
Information	11/11/2022 10:05:41 AM	Windows Defe...	2010	None
Warning	11/11/2022 10:05:29 AM	Windows Defe...	1121	None
Information	11/11/2022 10:05:22 AM	Windows Defe...	2010	None
Information	11/11/2022 9:51:49 AM	Windows Defe...	2010	None

Event 1121, Windows Defender

General Details

```
Microsoft Defender Exploit Guard has blocked an operation that is not allowed by your IT administrator.
 For more information please contact your IT administrator.
        ID: 9E6C4E1F-7D60-472F-BA1A-A39EF669E4B2
        Detection time: 2022-11-11T18:05:29.169Z
        User: [redacted]
        Path: C:\Users\[redacted]\AppData\Local\Temp\7zS0D7DB147\Installer.exe
        Process Name: C:\Windows\System32\lsass.exe
        Target Commandline: .\Installer.exe /silent
        Parent Commandline:
        Involved File:
        Inheritance Flags: 0x00000000
        Security intelligence Version: 1.379.201.0
        Engine Version: 1.1.19800.4
        Product Version: 4.18.2210.6
```

Log Name:	Microsoft-Windows-Windows Defender/Operational		
Source:	Windows Defender	Logged:	11/11/2022 10:05:29 AM
Event ID:	1121	Task Category:	None
Level:	Warning	Keywords:	
User:	SYSTEM	Computer:	B[redacted]
OpCode:	Info		
More Information:	Event Log Online Help		

Figure 3.3 – ASR block ETW event 1121

The generated event will allow you to observe what was blocked and which rule GUID was matched. Since ASR rules can be modified via security intelligence updates, the version of security intelligence is also relevant.

Warn mode

Warn mode provides a break-the-glass scenario for ASR. When configured in this mode, the rule blocks the targeted activity and produces a block ETW event, just as with **block** mode. However, the pop-up toast notification for **warn** mode differs from that of **block** mode. For **warn** mode, the toast notification provides users an option to unblock and exclude the underlying activity for 24 hours. Upon clicking on the **Unblock** button on the pop-up toast notification, all successive occurrences of the underlying activity are excluded for 24 hours. After 24 hours, the temporary exclusion is removed, the activity is blocked again and the unblock option is provided again. If the rule in **warn** mode blocks multiple activities, acting on one pop-up toast notification will not automatically exclude the rest. Every blocked activity that needs to be excluded will need the **Unblock** button on the associated toast notification to be clicked upon.

Note

The default state of 15 out of 17 rules is **not configured**. The default state of the **Block abuse of exploited vulnerable signed drivers** rule and the **Block credential stealing from the Windows local security authority subsystem (lsass.exe)** rule is **configured**, and the default mode is **block** mode.

For Windows versions that support ASR but don't support **warn** mode, a rule configured in **warn** mode behaves exactly like one being enabled in **block** mode.

Exclusions

ASR rules have been designed to block certain behavior of the monitored actor(s) or against the monitored target(s). As such, for most of the rules, there is always a nonzero probability of blocking an unintended process or a scenario. The use of the phrase *unintended process or scenario* here is very deliberate—these are blocks that organizations don't want to be blocked, and they are generally third-party LOB apps and first-party internal tools, or apps installed on ASR-protected devices.

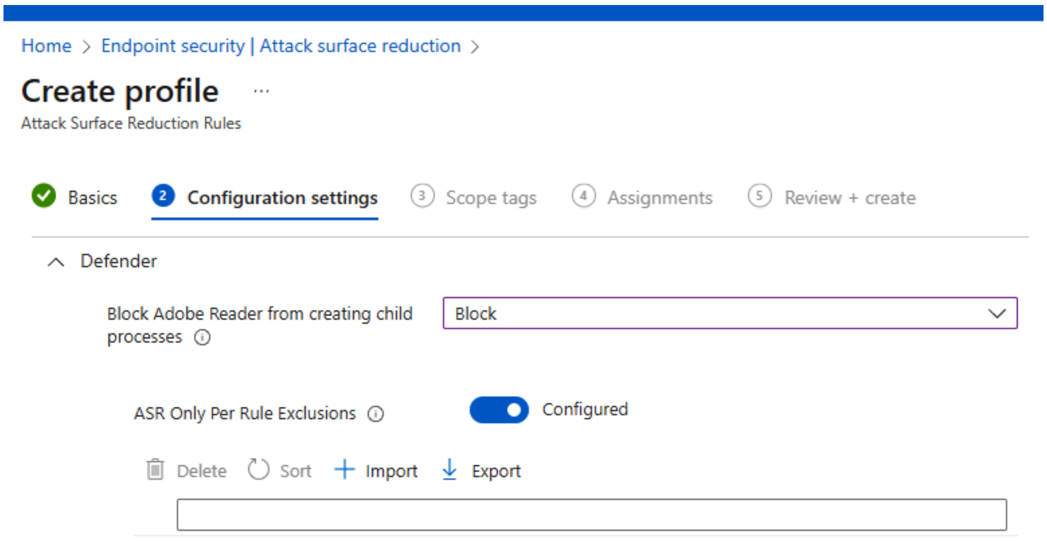
Every organization has its own set of applications that qualify for being deemed as unintended blocks and successively as FPs. Also, it is noteworthy that what is considered an FP by one organization may or may not be considered an FP by other organizations. In fact, it may not even qualify as an FP on a different group of devices within the same org. Furthermore, it could even be considered a **true positive** (**TP**) for a certain device group in a certain organization.

Due to the dynamic and subjective nature of FPs, ASR has been designed to be a self-serve feature—organizations can use **audit** mode to evaluate the impact of a particular rule and use one of the methods to exclude unintended blocks (actually, they are called audits when the rule is in **audit** mode since the action is not really blocked).

File and folder path exclusions

File and folder path exclusions are defined using the path of the blocked target. However, there is one exception—in the case of the **Block credential stealing from the Windows local security authority subsystem** rule, the target is always `lsass.exe` (since the rule has been defined to protect the target) and the exclusions are not really required (more details in the *Rule categories and descriptions* section earlier in this chapter). However, in case an organization still decides to exclude a block, file and folder path exclusions are defined using the file path or the folder path of the parent process blocked from accessing `lsass.exe` process memory.

Once created, matching files or files within the matching folders are no longer blocked. In addition, exclusions can now be configured as rule-specific! In the following screenshot, you'll see that you can enable **ASR Only Per Rule Exclusions**:



Block Adobe Reader from creating child processes: Block

ASR Only Per Rule Exclusions: Configured

Figure 3.4 – Per-rule exclusion example

ASR supports wildcards (`*`) as well as all standard environment variables when creating file and folder path exclusions. Some examples of environment variables you might use:

- `%appdata%`
- `%localappdata%`
- `%userprofile%`
- `%temp%`
- `%programfiles%`
- `%programfiles(x86)%`
- `%windir%`

An ASR rule-specific list of example blocks and suggested file and folder exclusions for handling the respective blocks can be found in *Chapter 10, Reference Guide, Tips, and Tricks*.

Allow indicator exclusions

ASR also honors *allow file* and *allow certificate* indicators. No file or process matching with any of the *allow file hash* IoCs or the *allow certificate* IoCs is blocked by any of the ASR rules. However, as with file and folder path exclusions, these are not rule-specific exclusions. Once defined, they are honored by all rules alike.

The prerequisites for using indicator exclusions are:

- **Microsoft Defender Antivirus** (**Defender Antivirus**) must be in **active** mode
- Cloud-based protection must be enabled
- The antimalware client must be up to date (`4.18.1901.x` or later)
- You must be using a supported OS version. Supported Windows versions are the following:
  - Windows 10, version 1703 or later
  - Windows Server 2012, 2012 R2, 2016, 2019, and 2022 are all supported, though note that Windows Server 2016 and Windows Server 2012 R2 must be onboarded using the modern, unified solution released in 2022.

Cold snack

All Defender Antivirus exclusions are honored by ASR by default. Any process, file, or folder excluded using Defender exclusions is not blocked by any of the ASR rules—there's no reason to create ASR-specific exclusions.

Analyzing ASR telemetry using AH

MDE captures every ASR event along with rich sets of information. This data is available to be accessed using AH. However, it is important to note that ASR events are deduplicated every hour across the entire organization. Hence, only the first event that occurred for a given rule and process combination within the entire organization in each hour is reported in AH.

The ASR telemetry is accessed through AH via the **ActionTypes** column of the `DeviceEvents` table. Here is a list of action types for reference:

AH ASR action type	AuditMode	BlockMode
Block Adobe Reader from creating child processes	`AsrAdobeReader ChildProcessAudited`	`AsrAdobeReader ChildProcessBlocked`
Block executable content from email client and webmail	`AsrExecutable EmailContentAudited`	`AsrExecutableEmail ContentBlocked`
Block Office applications from creating executable content	`AsrExecutableOffice ContentAudited`	`AsrExecutableOffice ContentBlocked`
Block credential stealing from the Windows local security authority subsystem	`AsrLsassCredential TheftAudited`	`AsrLsassCredential TheftBlocked`
Block execution of potentially obfuscated scripts	`AsrObfuscated ScriptAudited`	`AsrObfuscated ScriptBlocked`
Block all Office applications from creating child processes	`AsrOfficeChild ProcessAudited`	`AsrOfficeChild ProcessBlocked`
Block Office communication application from creating child processes	`AsrOfficeCommApp ChildProcessAudited`	`AsrOfficeCommApp ChildProcessBlocked`
Block Win32 API calls from Office macros	`AsrOfficeMacroWin32 ApiCallsAudited`	`AsrOfficeMacroWin32 ApiCallsBlocked`

AH ASR action type	AuditMode	BlockMode
Block Office applications from injecting code into other processes	`AsrOfficeProcessInjectionAudited`	`AsrOfficeProcessInjectionBlocked`
Block persistence through WMI event subscription	`AsrPersistenceThroughWmiAudited`	`AsrPersistenceThroughWmiBlocked`
Block process creations originating from PsExec and WMI commands	`AsrPsexecWmiChildProcessAudited`	`AsrPsexecWmiChildProcessBlocked`
Use advanced protection against ransomware	`AsrRansomwareAudited`	`AsrRansomwareBlocked`
Block JavaScript or VBScript from launching downloaded executable content	`AsrScriptExecutableDownloadAudited`	`AsrScriptExecutableDownloadBlocked`
Block executable files from running unless they meet a prevalence, age, or trusted list criterion	`AsrUntrustedExecutableAudited`	`AsrUntrustedExecutableBlocked`
Block untrusted and unsigned processes that run from USB	`AsrUntrustedUsbProcessAudited`	`AsrUntrustedUsbProcessBlocked`
Block abuse of exploited vulnerable signed drivers	`AsrVulnerableSignedDriverAudited`	`AsrVulnerableSignedDriverBlocked`

Table 3.2 – Action types and ASR rule events in AH

When browsing through ASR telemetry using AH, the following queries could be very useful, to enlist the unique device count and total event count for each ASR action type. However, please note that since these figures are produced on top of the deduplicated data, the original count of unique devices and total events could be much higher for most of the rules:

```
DeviceEvents
| where ActionType startswith ‹Asr›
| distinct ActionType, DeviceId
| summarize deviceCount = count() by ActionType
| join (DeviceEvents | where ActionType startswith 'Asr'
| summarize ruleCount = count() by ActionType) on $left.
ActionType == $right.ActionType
| project ActionType, deviceCount, ruleCount
```

To produce the `FolderPath` and `FileName` breakup for a specific ASR action type, again, the emphasis would be on the previous comment about aggregation on top of deduplicated data:

```
DeviceEvents
| where ActionType == ‹AsrVulnerableSignedDriverBlocked'
| distinct FolderPath, FileName, DeviceId
| summarize deviceCount = count() by FolderPath, FileName
| join (DeviceEvents| where ActionType ==
"AsrVulnerableSignedDriverBlocked"| summarize ruleCount =
count() by FolderPath, FileName) on $left.FolderPath == $right.
FolderPath and $left.FileName == $right.FileName
| project FolderPath, FileName, deviceCount, ruleCount
| order by ruleCount desc
```

Here's the code to produce the `FolderPath` and `FileName` breakup for all ActionTypes seen in the telemetry:

```
DeviceEvents
//| where ActionType == ‹AsrPsexecWmiChildProcessAudited'
| where ActionType startswith ‹Asr›
| distinct ActionType, FolderPath, FileName, DeviceId
| summarize deviceCount = count() by ActionType, FolderPath,
FileName
| join (DeviceEvents| summarize ruleCount = count() by
ActionType, FolderPath, FileName) on $left.ActionType ==
$right.ActionType and $left.FolderPath == $right.FolderPath and
$left.FileName == $right.FileName
```

```
| project ActionType, FolderPath, FileName, deviceCount,
ruleCount
| order by ActionType, ruleCount desc
```

Now that we have covered the logical groupings of ASR rules, the intent, and what each does, let us move into other areas of ASR.

Network protection layers and controls

Let's dive into **network protection** (**NP**), another feature under the MDE umbrella, specifically within the ASR space. Before we get into it all the wonderful things it can do, let's talk a little bit about its history and where it started out!

NP was not Windows Defender's original attempt at this type of protection—that was the **network resource inspection/network inspection system** (**NRI/NIS**). This was a very powerful feature that allowed our researchers to leverage custom protocol parsers and signatures; the issue was that it was a bit unreliable and seemingly overcomplicated.

Cold snack

Some of these parsers failed routinely, each failure sending a **Microsoft Active Protection Service** (**MAPS**) report. At one point, a particular failure was the cause of nearly 40% of all MAPS traffic. Let's just say, that was expensive.

Around the arrival of Windows 10 version 1709, or RS3, the NP team and the **Microsoft SmartScreen** (**SmartScreen**) team merged. From there, the conversation started: how can we take SmartScreen and extend it to other browsers? One of the initial thoughts was to take the core of SmartScreen and pair it with a network filter, scraping the **generic application-level protocol** (**GAPA**) engine-based inspection that was originally being used.

When Windows 10 version 1803 (RS4) came out, the two products were merged into one, and the underlying engine was switched out from what was called GAPA (as mentioned earlier) for the **malware protection engine** (**MPEngine**), which then drove the signatures going forward.

Alright—now that we know a little more about the journey NP took, let's talk a little bit about what it can do today. Currently, it supports the blocking and auditing of HTTP, UDP, FTP, SSH, RDP, and HTTPS/TLS/SSL (encrypted connections can only be parsed to extract the hostname, not the full URL, which of course lowers the granularity), which includes both inbound and outbound connections, as well as the ability to inspect UDP and DNS traffic.

In *Figure 3.5*, we see what the NP flow looks like, and at face value, it should be viewed with the **endpoint detection and response (EDR) Sensor** (**Sense**) component out of the picture, which comes next.

One of the core features of NP when it comes to MDE is the use of **custom indicators**, blocking things such as URLs and IP addresses. NP is ultimately blocking what the SmartScreen service thinks has a bad reputation, so it's the SmartScreen service making the decision. However, for custom indicators, the MDE backend is sending those instructions. Sense only technically receives data from NP; it does not instruct NP to do anything. Let's look at that a little closer here:

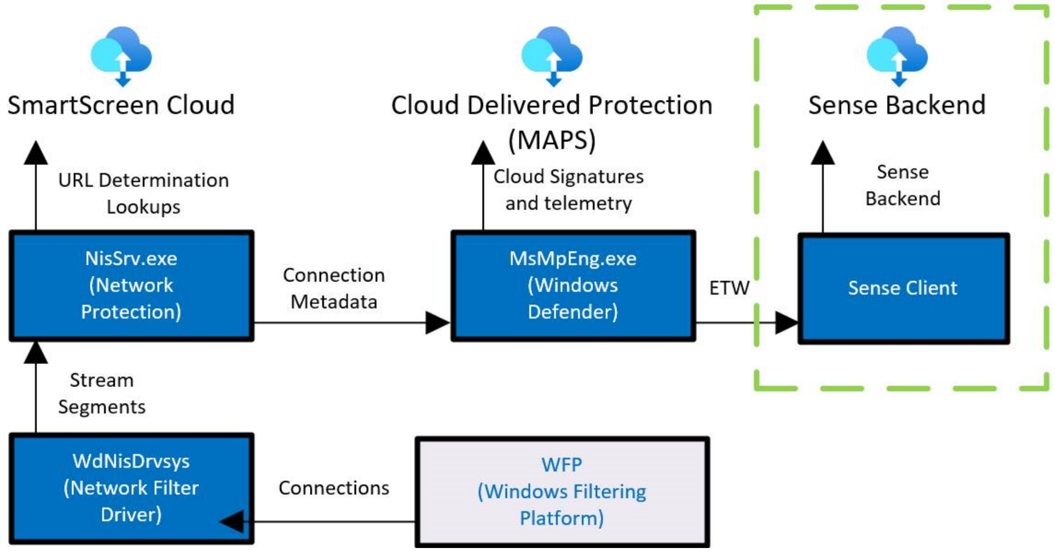

SmartScreen Cloud
URL Determination Lookups
Cloud Delivered Protection (MAPS)
Cloud Signatures and telemetry
Sense Backend
Sense Backend
NisSrv.exe (Network Protection)
Connection Metadata
MsMpEng.exe (Windows Defender)
ETW
Sense Client
Stream Segments
WdNisDrvsys (Network Filter Driver)
Connections
WFP (Windows Filtering Platform)

Figure 3.5 – Network protection flow for blocking indicators

As we can see from the preceding diagram, `NisSrv` (which is what we call Network Protection) does the parsing and then relays that to the SmartScreen cloud service for analysis. We also then get MPEngine doing the signature matching with **cloud-delivered protection** (**CDP**).

Some other items to share about Network Protection would be around the telemetry that it sends to the M365D portal, things that you can gather in AH. That includes information such as **Port Open**, **Connection Established**, **Reputation results**, and **Connection volumes**.

Custom indicators

Custom indicators are one of the more interesting features when it comes to the power of NP. This is where we can specify things such as IP addresses and URLs/domains. Now, custom indicators also allow you to add things such as files and certificates as well, but those are not handled by Network Protection and will be covered in a later chapter. When adding IP addresses, URLs, or domains, you're setting whether you want those things allowed, audited, or blocked, or to warn the user when they or an application reaches out to them.

Cold snack

Network Protection also sends connection metadata to MPEngine for further behavioral monitoring. This is what used to be called **network real-time inspection** (**NRI**).

Operating modes

Let us talk about the different modes that NP can operate in. We will cover **log/inspect**, **audit**, and **block** modes.

Log or Inspect mode

The mode that is on by default for all devices is called **log/inspect**, or **disabled**. The reason this mode is called **disabled** in the public documentation is that while NP is disabled, the NIS/NRI component is still present and active. In this mode, it is only looking at untrusted processes, and if none is observed, then it will disable itself. In fact, for untrusted processes, it is sending metadata on untrusted processes for signature matching.

Audit mode

Next up is **audit** mode, which is the next level up. In this mode, all processes are monitored now. At this point, the network driver is still in read-only mode, and while it is contacting the SmartScreen service, it's only to do reputation checks. These events are also written in the event log, as shown in the following table:

Event ID	Provider/source	Description
`5007`	Windows Defender (Operational)	Event when settings are changed
`1125`	Windows Defender (Operational)	Event when a network connection is audited
`1126`	Windows Defender (Operational)	Event when a network connection is blocked

Table 3.3 – Network protection event IDs

Block mode

Finally, **block** mode. Again, here, all processes are monitored, but now the network driver is in blocking mode, which means packets are held until a determination can be made. Let us look at the blocking decision a little deeper:

1. The end user or an application attempts a connection to a **Uniform Resource Identifier** (**URI**).
2. The connection is inspected and parsed for the URI information.
3. The URI is checked against our Bloom filters, and if it hits on that, it will continue through the logic chain. If it does not, the connection is released and allowed to continue.

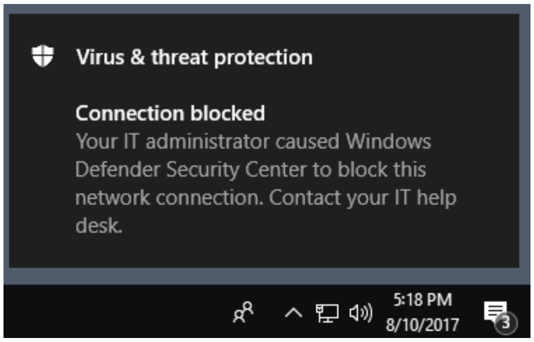
4. From here, the SmartScreen service is queried to give its determination of the URI.
5. If the determination is block or warn, the following will happen:

   A. A block is injected into the network stream.

   B. Events are logged, as shown in *Table 3.3*.

   C. The event is sent to Windows Defender (see *Figure 3.6*), and then reported to Sense.

This decision then gets recorded locally:



Virus & threat protection

Connection blocked

Your IT administrator caused Windows Defender Security Center to block this network connection. Contact your IT help desk.

5:18 PM
8/10/2017

Figure 3.6 – NP block toast notification

Knowing about the TCP three-way handshake is important to understanding this flow as well, especially when viewing alerts or AH data regarding these allows or blocks: when a connection is initiated, you will see the handshake process generate a connection success first. To match the connection attempt against an indicator, the handshake must first complete. However, no traffic is allowed after a block!

Cold snack

Network Protection updates monthly via **monthly common antimalware platform** (**MoCAMP**) updates.

Where can NP be used?

- Windows, of course!
- MacOS
- Linux
- Android and IOS

We'll cover more of the data that gets sent to the Defender portal in *Chapter 8*, *Establishing Security Operations*, as we look to cover more operations-related work. That wraps up Network Protection for now, so let's move into CFA.

CFA ransomware mitigations

CFA is yet another feature in the ASR space that leverages the Defender engine to harden devices against ransomware and other destructive apps or threats. The whole concept of CFA is to control which apps are allowed to access what are marked as protected folders.

Operating modes

CFA has been designed to operate on unfriendly and untrusted processes. These are processes that are not validly signed and, based on Microsoft's **cloud reputation system**, not known to be, or not already determined as, clean or friendly processes. CFA provides multiple levels of blocking and auditing for the activities performed by such processes in the form of various operational modes.

Disable (default)

In this mode, the feature is disabled and won't block any activity from any process. By default, CFA is disabled and needs to be explicitly enabled in one of the non-disabled modes to block or audit the activity.

Enable (block)

In this mode, CFA ensures that unfriendly and untrusted processes are not allowed to make changes to the files in *protected folders*. The following folders are included by default and cannot be removed from the list. You can add more folders to the list. All such folders will be protected by CFA:

- `%userprofile%\Documents`
- `%userprofile%\Pictures`
- `%userprofile%\Videos`
- `%userprofile%\Music`
- `%userprofile%\Favorites`
- `c:\Users\Public\Documents`
- `c:\Users\Public\Pictures`
- `c:\Users\Public\Videos`
- `c:\Users\Public\Music`

Audit mode

This is the evaluation mode for CFA for protected folders. When configured in this mode, CFA will audit all modifications to files within protected folders. This mode is particularly useful in determining all unfriendly and untrusted processes that will be blocked by CFA. This provides you the opportunity to review the expected impact and add exclusions wherever needed before enabling CFA in **block** mode.

Block disk modification only

This mode allows CFA to block all unfriendly and untrusted processes from making modifications to the disk sectors. Some malware is known to encrypt data by encrypting disk sectors. In this mode, CFA blocks all disk sector modification attempts from such malware, including things such as the **Master Boot Record** (**MBR**).

Audit disk modification only

As with **audit** mode for protected folders, this is another evaluation mode for CFA. In this mode, CFA audits all disk sector modifications by unfriendly and untrusted processes. No process is blocked. This auditing allows users to review the expected impact and deploy exclusions before enabling the **Block disk modification only** setting.

Story from the field

There was a blog written collectively in January 2022 by the **Microsoft Threat Intelligence Center** (**MSTIC**), **Microsoft Digital Security Unit** (**DSU**), **Microsoft 365 Defender Threat Intelligence team**, and **Detection and Response Team** (**DART**) surrounding the situation in Ukraine. This covered things specific to the malware referred to as **Wiper** that was making its way around. This was particularly nasty because, unlike most malware, there was no ransomware aspect. It seemed to be created only to be destructive. In essence, it aimed to overwrite the MBR with a ransom note. Granted—in this case, there is no actual recovery situation; the drive is corrupted.

As explained with the **Block disk modification only** setting, this would have greatly helped protect against an attack such as this. As threats evolve, more and more protections within the MDE suite need to be considered, even though some can take some intensive planning or preparation—all things we aim to help with in this book.

If you'd like to read more on Wiper, Microsoft published the *Destructive malware targeting Ukrainian organizations* blog post covering the details. It can be found at `https://www.microsoft.com/security/blog/2022/01/15/destructive-malware-targeting-ukrainian-organizations/`.

Exploit protection for advanced mitigations

Exploit protection (**EP**) was introduced in Windows 10 to integrate the **enhanced experience mitigation experience toolkit** (**EMET**) functionality. Many of the mitigations were simply incorporated into the operating system or enabled by default, as Windows 10 came with higher standards for application security and with its own mitigations against **return-oriented programming** (**ROP**). Exploit Protection's key value comes from providing a way to mitigate against known **vulnerabilities** in older (pre-Windows 10) software or to make exceptions for them.

EP, as with traditional HIPS, requires knowledge of the operating system and the vulnerable software to ensure you get it right: consequently, this complexity increases the risk of impacting performance or functionality.

The best way of understanding the application of this technology is by viewing it as mitigation against specific vulnerabilities when a piece of software was not written using the security capabilities available in the operating system. Typically, this applies to legacy applications most of all.

Finally, several other capabilities provided, often in a more robust and less complex way, protect against the exploitation of known vulnerabilities as well. They should be the first stop on the way toward mitigation:

- **Threat vulnerability management** (**TVM**): The best way to identify and remediate known vulnerabilities. Very often, a software or operating system update is the best possible mitigation.
- **Defender Antivirus**: (Also, see *Chapter 2*, *Exploring Next-Generation Protection*) provides in-memory scanning and protection, including behavior monitoring that can block known malicious behavior including techniques that attempt to exploit vulnerabilities, even against unknown ones. There's a ton of research that goes into this every day, augmenting your ability to defend and making a lot of precise, specific, and hard-to-maintain exploit protection mitigations redundant.
- **ASR rules**: Just as with behavior monitoring but in a more tangible/visible way, these address common attack vectors and protect against a very large set of possible techniques but with much less complexity or prerequisite knowledge: also, a result of the large amount of research that goes into these every day.

Now that you have also added a solid understanding of EP to your toolbelt, let's reflect on what we've learned.

Summary

In this chapter, we covered ASR rules, NP and CFA and their operating modes, and finally, exploit protection. From that, you've learned which options there are and are armed with enough best-practice information to work through implementation in your own network, focusing on what will be most effective, while avoiding potential impact where possible.

In the next chapter, we'll discuss EDR features within MDE, laying a foundational understanding that we'll leverage for practical application in *Chapter 8*, *Establishing Security Operations*.