



1ST EDITION

Microsoft Defender for Identity in Depth

An exhaustive guide to ITDR, breach prevention,
and cyberattack response



PIERRE THOOR

Foreword by Matthew Zorich, Principal Security Research Manager, Microsoft GHOST



Microsoft Defender for Identity in Depth

Copyright © 2024 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

The author acknowledges the use of cutting-edge AI, such as ChatGPT/Claude/Grammarly, with the sole aim of enhancing the language and clarity within the book, thereby ensuring a smooth reading experience for readers. It's important to note that the content itself has been crafted by the author and edited by a professional publishing team.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

Group Product Manager : Dhruv Jagdish Kataria

Publishing Product Manager : Prachi Sawant

Book Project Manager : Ashwin Kharwa

Senior Editor : Sarada Biswas

Technical Editor : Rajat Sharma

Copy Editor : Safis Editing

Proofreader : Sarada Biswas

Indexer : Pratik Shirodkar

Production Designer : Jyoti Kadam

Senior DevRel Marketing Executive : Marylou De Mello

First published: December 2024

Production reference: 1251124

Published by Packt Publishing Ltd.

Grosvenor House

11 St Paul's Square

Birmingham

B3 1RB, UK.

ISBN 978-1-83588-448-5

www.packtpub.com

*For Carina, the love of my life, and our amazing children – your support and love mean the world
to me.*

- Pierre Thoor

Foreword

A well-known phrase in cyber security is “*hackers don’t break in, they log in.*” Anyone who has worked in incident response, in a security operations center, or in any kind of threat-hunting role can attest to the truth of this statement. Despite being with us for many years, and despite the evolution of cloud identity platforms such as Microsoft Entra ID, in many organizations, Microsoft Active Directory is still at the heart of the corporate identity landscape. Identities, from our regular users to our most privileged administrators, are key to maintaining the security of our environments or limiting the impact of compromise. I have been fortunate to be involved in some of the largest and most complex cybersecurity breaches in the world and have seen defenders struggle to protect Active Directory effectively. It is understandable that defenders would struggle; modern identity systems are complex and ever-changing to suit the evolving needs of modern companies. The advent of the work-from-home era has increased this complexity significantly and further highlighted the importance of securing our identities. This evolving complexity presents a unique challenge to defenders. It is difficult to collect the logs and telemetry from all these services manually and build custom detection rules across them all to effectively secure your identity plane. **Microsoft Defender for Identity (MDI)** looks to solve this problem by reducing the barrier to entry to be able to effectively monitor Active Directory. The out-of-the-box telemetry and detection logic has been tuned over years of understanding how adversaries compromise Active Directory, taken from the lessons from real-world compromises. As new techniques are discovered, they are built and deployed into the product, ensuring protection against the most novel of attacks.

MDI is one of the pillars of the Microsoft Defender security stack. It provides unique visibility and insights into not only Active Directory, but also other supporting components, such as Active Directory Federation Services, Active Directory Certificate Services, and Microsoft Entra Connect. My personal belief is there is no product available that comes close to providing the out-of-the-box insights that MDI does. I have often joked that if I could spend someone else’s money on a security product, then MDI would be it. In this book, you will learn from Pierre, an expert in the deployment and operationalization of MDI. Importantly, he will arm you with the knowledge to not only configure and deploy MDI but also to empower you with the skills to effectively hunt for and prevent identity compromise. This book covers not only reactive investigations but, crucially for defenders, how to use MDI to proactively address misconfigurations or weaknesses in Active Directory. The goal of any threat actor is to obtain an identity with enough privilege to complete their objectives, from data exfiltration to ransomware and everything in between. This book will arm you with the skills to hopefully prevent, disrupt, or understand that activity in your environment.

Matthew Zorich

Principal Security Research Manager

Microsoft GHOST

Contributors

About the author

Pierre Thoor is a Microsoft MVP in security and a dedicated cybersecurity expert with a focus on identity protection and threat detection. As a first-time author, he shares his extensive knowledge in this book. Pierre hosts the Security Dojo Podcast and blogs at thoor.tech, where he explores Microsoft security topics. As an international speaker, he makes complex security subjects accessible to audiences worldwide.

At Onevinn, Pierre delivers advanced security solutions that strengthen organizations' defenses against cyber threats. He specializes in Microsoft Sentinel and Microsoft Defender XDR. Pierre is also an expert in Azure Governance, including the Cloud Adoption Framework and enterprise-scale landing zones, ensuring that security is integrated into every aspect of cloud adoption. With skills in DevOps practices, **Kusto Query Language (KQL)**, and developing solutions with Bicep and PowerShell, he implements automation and infrastructure as code to enhance security operations.

Pierre assists organizations in navigating the complexities of modern cybersecurity challenges.

I want to deeply thank my wife, Carina, for her constant support and belief in me. To my children, whose curiosity and questions inspired me – thank you. Big thanks to the technical reviewers, Stefan Schörling and Konrad Sagala, for your helpful support and comments that made this book better. Thank you, Matthew Zorich, for your endless inspiration and for writing the foreword.

Lastly, thank you to my employer, Onevinn, for your support during this journey.

About the reviewers

Stefan Schörling is a renowned security expert with more than 25 years of experience in the cybersecurity field. He has served various roles within the cybersecurity area. Today, he is supporting customers to be better protected against adversaries, but also helping those who have been hit by cyber incidents. In his spare time, he conducts security research and also speaks at various international conferences (SANS, Live 360, TechED, Ignite, and various user groups). Stefan has been awarded a Microsoft MVP award every year for 17 years for his efforts and passion for sharing his knowledge in tech communities.

I'd like to thank my wife and family for the time and commitment it takes for me to follow my dream and do the things I do. Without their support, this would not be possible. I would also like to thank my first manager, Mathias, for believing in me even if I had no formal IT education and for the fun times we had over the years working together. I would simply not be where I am today without them.

Konrad Sagala has been involved in designing and deploying server systems since 1993. From 1996, he focused on Windows Server Systems: Security, Exchange and Active Directory. For the last 10 years, he has focused on cloud platforms. He has been an active Microsoft Certified Trainer since 2007, delivering Identity, Microsoft 365, Exchange, Security, and Server Platform courses, and has been a Microsoft Most Valuable Professional for 18 years in the Microsoft 365 category.

Table of Contents

Preface

Part 1: Mastering the Fundamentals of Microsoft Defender for Identity

1

Introduction to Microsoft Defender for Identity

The growing threat landscape and the role of MDI in ITDR

Modern identity threats and strategic defense frameworks

The Cyber Kill Chain

MITRE ATT&CK framework

The Unified Kill Chain

MDI's strategic position in the cybersecurity ecosystem

Unpacking key features and benefits of MDI

Summary

2

Setting up Microsoft Defender for Identity

Technical requirements

Pre-installation and planning checklist: laying the groundwork

Licensing

What permissions do you need?

What are the operating system requirements?

Other sensor requirements

Networking

PowerShell

Data collection

User profiling

Sizing

Prerequisites for AD FS and AD CS

Active Directory service accounts

Deployment of MDI – a step-by-step guide

Following with your own lab environment

Getting the MDI installation package and access key

Navigating step-by-step proxy configuration for MDI

Installing TinyProxy

Configuring TinyProxy

Ensuring success with post-installation activities

DSAs

Configuring SAM-R

Setting the gMSA in the Defender XDR portal

Verifying the DSA

Defender XDR unified RBAC

Summary

Leveraging MDI PowerShell for Automation and Management

Technical requirements

Primer on the MDI PowerShell module

Installing the MDI PowerShell module

Module file overview

Understanding the module and its functions

Crafting advanced PowerShell scripts for MDI management

Health issues API

Automation in action – case studies and scripting scenarios

Monitoring the MDI service via Azure Monitor

Monitoring the MDI configuration with Azure Monitor and custom alert rules

Sending health issues and security alerts via syslog to Microsoft Sentinel

Summary

Part 2: Advanced Configuration, Integration, and Threat Detection

4

Integrating MDI with AD FS, AD CS, and Entra Connect

Technical requirements

Integrating MDI with AD FS

How AD FS authentication works

Configuring AD FS for MDI sensor installation

Validating the AD FS integration

Integrating MDI with AD CS

How AD CS works

Importance of MDI on Certificate Servers

Configuring AD CS for MDI sensor installation

Validating the AD CS integration

Integrating MDI with Entra Connect

How Entra Connect works

Configuring Entra Connect for the MDI sensor

Validating the Entra Connect integration

Expanding MDI across multiple Active Directory forests

The concept of multiple forests

Types of trusts in multi-forest environments

Prerequisites for MDI in a multi-forest environment

VPN integration – securing remote activities and data flow

Understanding RRAS and RADIUS

Configuring Microsoft RRAS

Summary

5

Extending MDI Capabilities Through APIs

Technical requirements

Introduction to the MDI API

Getting started with Microsoft Graph API

Building custom integrations and automations

Identifying integration opportunities

Type of use cases

Summary

6

Mastering KQL for Advanced Threat Detection in MDI

Technical requirements

KQL for beginners – querying MDI data

The history of KQL and its ecosystem

Understanding your MDI data

Getting started with KQL

Practical tips for effective queries

Hunting tables in MDI

Practical use of hunting tables

Advanced KQL techniques for deep threat detection

Understanding attack paths in AD

MDI and the attacker's kill chain

Crafting KQL queries for threat detection

Real-world case studies – detecting advanced attacks with KQL

Prerequisites

PtH attack

Kerberoasting

DCShadow attack

Summary

Further reading

Part 3: Operational Excellence with Microsoft Defender for Identity

7

Investigating and Responding to Security Alerts

Developing a methodical approach to alert investigation

Understanding the MDI alert system

User Entity

Lateral movement paths (LMPs)

Initial triage and categorization

Root cause analysis

Real-world playbook – responding to advanced threats

Defining advanced threats

Pre-incident preparation

Incident detection and validation

Response strategy and execution

Incident response – an action plan for high-stakes situations

Building an incident response team

Incident Response Plan (IRP)

Summary

8

Utilizing MDI Action Accounts Effectively

Technical requirements

Configuring and securing action accounts

Understanding action accounts – what are they and why do they matter?

Best practices for action account configuration – getting it right the first time

Security measures – protecting your action accounts from compromise

Real-world scenarios and use cases

Automated threat response – leveraging action accounts for quick reactions

Case study – detecting and responding to credential theft and lateral movement

Operational efficiency – how action accounts streamline security processes

Summary

9

Building a Resilient Identity Threat Detection and Response Framework

Technical requirements

Designing proactive threat-hunting strategies with MDI

Understanding the threat-hunting methodology

The importance of logging and accurate detection

Developing security use cases

Leveraging behavioral analytics and MDI in hypothesis-driven hunting

Elevating your ITDR posture – Continuous improvement with MDI

Learning from total identity compromise incidents

Implementing identity-hardening strategies

Disaster recovery and incident response – preparing for the inevitable

Establishing an incident response plan

Automating responses to identity-based incidents with SOAR

Disaster recovery for identity systems

Summary

10

Navigating Challenges: MDI Troubleshooting and Optimization

Diagnosing common MDI issues

Spotting the signs of trouble

Using tools and logs to find problems

Configuration and connectivity fixes

Checking key configuration settings

Removing a malfunctioning MDI sensor manually

Network connectivity troubleshooting

Resolving security alert misfires

Customizing detection rules and applying filtering techniques

[Adjusting alert settings for better accuracy](#)

[Operational guide](#)

[Daily tasks](#)

[Weekly tasks](#)

[Monthly tasks](#)

[Quarterly/ad-hoc tasks](#)

[Summary](#)

[Future reading](#)

[Index](#)

[Other Books You May Enjoy](#)

Preface

Welcome to a journey where we turn the tables on cyber adversaries using Microsoft Defender for Identity!

In today's digital landscape, cyber threats are becoming more sophisticated, targeting the very identities that form the backbone of our organizations. Microsoft **Defender for Identity (MDI)** is a powerful tool designed to help you protect your Active Directory environments from these advanced attacks. By providing deep insights into user activities and behaviors, MDI enables you to detect, investigate, and respond to threats effectively.

This book, *Microsoft Defender for Identity in Depth*, is your comprehensive guide to mastering MDI. It brings together everything you need in one place, from setting up and configuring MDI to exploring advanced features and integrations. Through hands-on examples and real-world scenarios, you'll learn how to secure identities, hunt for adversaries, and optimize your defenses against evolving cyber threats.

By diving deep into MDI, you'll enhance your ability to detect and catch adversaries, accelerate your learning curve, and position yourself to work confidently with other Microsoft security products. Whether you're starting from scratch or looking to refine your existing knowledge, this book will inspire you to elevate your cybersecurity skills to new heights.

Who this book is for

This book is designed for IT and security professionals eager to elevate their expertise in identity protection and threat management using Microsoft Defender for Identity. It is particularly valuable for the following:

- **System administrators** aiming to secure Active Directory environments
- **Cybersecurity analysts** seeking to enhance their detection and response capabilities
- **Identity and access management specialists** focused on safeguarding user identities
- **Incident response team members** who require effective tools for investigating security incidents
- **Threat hunters** interested in proactively identifying and mitigating risks
- **Cloud security engineers** looking to integrate MDI within broader security strategies

What this book covers

[**Chapter 1**](#), *Introduction to Microsoft Defender for Identity*, begins our journey by exploring the “why” behind MDI before diving into technical details. We examine its critical role within the evolving threat landscape and its place in **Identity Threat Detection and Response** (ITDR). By understanding modern identity threats, MDI’s strategic importance in cybersecurity, and unpacking its key features and benefits, you’ll gain a solid foundation on how MDI fortifies defenses against identity-centric attacks.

[**Chapter 2**](#), *Setting up Microsoft Defender for Identity*, guides you through securely deploying MDI. It includes a pre-installation checklist, step-by-step installation with proxy configurations, and post-installation verification to ensure MDI operates correctly.

[**Chapter 3**](#), *Leveraging MDI PowerShell for Automation and Management*, teaches you how to use the MDI PowerShell module to automate and manage Microsoft Defender for Identity. You’ll learn about key commands and automation techniques to streamline MDI administration.

[**Chapter 4**](#), *Integrating MDI with AD FS, AD CS, and Entra Connect*, teaches you how to integrate Microsoft Defender for Identity with key Active Directory services: **Active Directory Federation Services** (AD FS), **Active Directory Certificate Services** (AD CS), and Entra Connect. You’ll learn how to expand MDI’s coverage across multiple Active Directory forests and integrate with VPNs to secure remote activities.

[**Chapter 5**](#), *Extending MDI Capabilities Through APIs*, explores how to extend Microsoft Defender for Identity using the Microsoft Graph API. You’ll focus on key APIs that allow you to monitor and manage alerts, incidents, and health issues within your MDI environment.

[**Chapter 6**](#), *Mastering KQL for Advanced Threat Detection in MDI*, teaches you how to use **Kusto Query Language** (KQL) within Microsoft Defender for Identity to enhance your threat detection capabilities. You’ll start with the basics of querying and filtering MDI data, then progress to advanced techniques for identifying hidden patterns and anomalies. Through real-world case studies, you’ll learn how to detect advanced attacks, empowering you to improve your organization’s security defenses.

[**Chapter 7**](#), *Investigating and Responding to Security Alerts*, teaches you how to effectively investigate and respond to security alerts in Microsoft Defender for Identity. You’ll establish a methodical approach for accurate threat identification and assessment. The chapter presents a real-world playbook for responding to advanced threats with swift action strategies. It also outlines a comprehensive incident response plan for high-stakes situations, preparing you to manage and mitigate security incidents effectively.

[Chapter 8](#), *Utilizing MDI Action Accounts Effectively*, delves into the strategic use of MDI action accounts. You'll learn how to configure them securely, following best practices to strengthen your security posture without introducing vulnerabilities. The chapter explores real-world scenarios and use cases, demonstrating how effectively managed action accounts play a pivotal role in automated threat response and operational efficiency within Microsoft Defender for Identity environments.

[Chapter 9](#), *Building a Resilient Identity Threat Detection and Response Framework*, focuses on constructing a robust ITDR framework using Microsoft Defender for Identity. You'll learn how to design proactive threat-hunting strategies with MDI, leveraging KQL to detect early indicators of compromise. The chapter discusses elevating your ITDR posture through continuous improvement and covers disaster recovery and incident response planning, preparing you for the inevitable challenges of security breaches.

[Chapter 10](#), *Navigating Challenges: MDI Troubleshooting and Optimization*, serves as a practical guide for IT professionals to troubleshoot and resolve common challenges with Microsoft Defender for Identity. You'll learn essential techniques for diagnosing and fixing frequent issues, including configuration and connectivity problems. The chapter delves into strategies for tuning performance and optimizing MDI operations to ensure a smooth and efficient security framework. It also provides insights into resolving false positives and alert misfires, enhancing the accuracy and reliability of your security measures.

NOTE

News from Microsoft Ignite 2024: Unified agent Microsoft has introduced a unified agent that integrates Microsoft Defender for Endpoint (MDE) with Microsoft Defender for Identity (MDI), extending protection across endpoints, operational technology (OT) devices, identities, and Data Loss Prevention (DLP). This consolidation simplifies deployment and maintenance by eliminating the need for separate agents, thereby reducing system overhead and enhancing efficiency. Organizations can now enable MDI directly from the Defender portal, streamlining the process of securing on-premises identities.

News from Microsoft Ignite 2024: Sensor Management API for Automated Operations To further enhance operational efficiency, Microsoft has launched a Sensor Management API. This API allows for the automation of tasks such as deployment, configuration, and monitoring of sensors within an organization's environment. By providing programmatic access, it enables security teams to maintain up-to-date sensor deployments and monitor their health status effectively, ensuring continuous and robust protection.

To get the most out of this book

This book assumes readers have foundational knowledge of Microsoft Defender for Identity, Kusto Query Language (KQL), Active Directory, and basic networking principles. Familiarity with the Microsoft 365 or Azure portals and experience with PowerShell will support you in following the labs and step-by-step instructions. For setup, ensure access to the specified software and environments in the following table.

Software/hardware covered in the book	Operating system requirements
Microsoft tenant	Windows, macOS, or Linux (to access the Microsoft or Azure portal)
Microsoft Defender for Identity subscription	
Azure subscription	Microsoft Azure
Active Directory, AD CS, AD FS, Entra Connect	Microsoft Windows Server 2019 or later
TinyProxy	Ubuntu
PowerShell 7.4 or later	Microsoft Windows Server 2019 or later

If you're familiar with infrastructure-as-code, you can use the Bicep templates provided in the GitHub repository to automate the deployment of the lab setup in your Azure environment.

If you are using the digital version of this book, we advise you to type the code yourself or access the code from the book's GitHub repository (a link is available in the next section). Doing so will help you avoid any potential errors related to the copying and pasting of code.

Download the example code files

You can download the example code files for this book from GitHub at

<https://github.com/PacktPublishing/Microsoft-Defender-for-Identity-in-Depth>. If there's an update to the code, it will be updated in the GitHub repository.

We also have other code bundles from our rich catalog of books and videos available at <https://github.com/PacktPublishing/>. Check them out!

Conventions used

There are a number of text conventions used throughout this book.

Code in text : Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. Here is an example: “A `.json` configuration setting file with connection information to your MDI instance.”

A block of code is set as follows:

```
$AccessKey = "<<Fill in the access key>>"  
Set-Location $PathForDestinationFiles  
. \"Azure ATP sensor Setup.exe" /quiet NetFrameworkCommandLineArguments="/q"  
AccessKey="$AccessKey"
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in **bold**:

```
Connect-MgGraph -Scopes "Application.ReadWrite.All"  
New-MgServicePrincipal -AppId <Application Id>
```

Any command-line input or output is written as follows:

```
sudo nano /etc/tinyproxy/tinyproxy.conf
```

Bold : Indicates a new term, an important word, or words that you see onscreen. For instance, words in menus or dialog boxes appear in **bold**. Here is an example: “Navigate to the **Sensor Installation Page**. You have two options to reach the **Add Sensor** page.”

TIPS OR IMPORTANT NOTES

Appear like this.

Get in touch

Feedback from our readers is always welcome.

General feedback : If you have questions about any aspect of this book, email us at customercare@packtpub.com and mention the book title in the subject of your message.

Errata : Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you would report this to us. Please visit www.packtpub.com/support/errata and fill in the form.

Piracy : If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at copyright@packtpub.com with a link to the material.

If you are interested in becoming an author : If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, please visit authors.packtpub.com.

Share Your Thoughts

Once you've read *Microsoft Defender for Identity in Depth*, we'd love to hear your thoughts! Please [click here to go straight to the Amazon review page](#) for this book and share your feedback.

Your review is important to us and the tech community and will help us make sure we're delivering excellent quality content.

Download a free PDF copy of this book

Thanks for purchasing this book!

Do you like to read on the go but are unable to carry your print books everywhere?

Is your eBook purchase not compatible with the device of your choice?

Don't worry, now with every Packt book you get a DRM-free PDF version of that book at no cost.

Read anywhere, any place, on any device. Search, copy, and paste code from your favorite technical books directly into your application.

The perks don't stop there, you can get exclusive access to discounts, newsletters, and great free content in your inbox daily

Follow these simple steps to get the benefits:

1. Scan the QR code or visit the link below



<https://packt.link/free-ebook/9781835884485>

2. Submit your proof of purchase
3. That's it! We'll send your free PDF and other benefits to your email directly

Part 1:Mastering the Fundamentals of Microsoft Defender for Identity

This part provides an essential foundation for understanding **Microsoft Defender for Identity (MDI)**. You'll be introduced to MDI's critical role in protecting against identity-based threats, guided through the deployment process, and equipped with automation techniques using PowerShell for streamlined management. This section is designed to give you a strong grasp of MDI's functionality and how to integrate it into your broader security strategy.

This part includes the following chapters:

- [Chapter 1](#) , *Introduction to Microsoft Defender for Identity*
- [Chapter 2](#) , *Setting up Microsoft Defender for Identity*
- [Chapter 3](#) , *Leveraging MDI PowerShell for Automation and Management*

1

Introduction to Microsoft Defender for Identity

In this starter chapter, we'll start boarding ourselves on the journey of **Microsoft Defender for Identity (MDI)** and its critical role within the evolving threat landscape. We'll get insights into the strategic importance of MDI within the broader cybersecurity ecosystem, learning how it serves as a fundamental tool in **Identity Threat Detection and Response (ITDR)**, which is a term from *Gartner*. This chapter will explore how MDI fits within the broader cybersecurity ecosystem, providing vital tools for protecting against identity-based threats.

Instead of jumping straight into the technical setup, we'll take time to explore the *why* behind MDI. Understanding these foundational aspects will give you a solid grasp of how MDI fits into a comprehensive security strategy. By the end of this chapter, you'll not only appreciate the capabilities of MDI but also how it helps to fortify your defenses against identity-centric attacks.

These insights are crucial for IT professionals and cybersecurity experts tasked with safeguarding **Active Directory (AD)** – a common target for adversaries. A successful attack on AD can lead to unauthorized access or even allow attackers to gain complete control over an environment, posing a severe threat to an organization's security and stability.

In this chapter, we will cover the following:

- The growing threat landscape and the role of MDI in ITDR
- Modern identity threats and strategic defense frameworks
- MDI's strategic position in the cybersecurity ecosystem
- Unpacking key features and benefits of MDI

Let's get started!

The growing threat landscape and the role of MDI in ITDR

As we begin this journey through the area of cybersecurity, we find ourselves navigating an ever-expanding threat landscape. The digital age, while bringing unparalleled convenience and connectivity, also introduces complex challenges that demand sophisticated solutions.

ITDR was identified by *Gartner Inc.* (and the term ITDR was created by them as well), an IT research and advisory company, as one of the top security and risk management trends that IT leaders and security leaders need to have a strategy on. Adversaries, attackers, hackers, we can call them whatever we want, abuse access and identities, and the focus of their attacks is identity compromise, lateral

movement, and privileged escalation. Therefore, we need tools and processes to detect, investigate, and respond to these types of threats to efficiently defend our organization. If we start thinking that ITDR is a security discipline and not just a product, to get visibility into credential abuse, privilege escalation attempts, and entitlement exposure, my opinion is that we then can know more about our environment and take appropriate actions for our security posture.

But what is ITDR? Before we answer that question, I want you to look at how our attack surface has expanded a lot in just a few years. Attackers are changing tactics and the spotlight on protecting our identities has never been so current as of now. While firewalls once served as our primary security boundary, the current landscape suggests that identity management is becoming a central element of security strategies. I believe this shift is driven by the rising numbers of password spray attacks, fundamental security misconfigurations – especially in implementing **multifactor authentication** (**MFA**) – and a lack of visibility into our data, leaving us remarkably vulnerable. Just before I began to write this book, Microsoft experienced an interesting nation-state attack from the group known as *Midnight Blizzard*, also referred to as *NOBELIUM*, famous for their *password-spray attacks* during 2021 against **Cloud Solution Providers (CSPs)** and **Managed Service Providers (MSPs)** and in January 2024 for their initial access through a password-spray attack of a legacy test OAuth application that had elevated privileged access.

Now, back to the question – what is ITDR? In this case, we are joining **Identity and Access Management (IAM)** together with **Extended Detection and Response (XDR)**. Many times, organizations are divided in the same way, where the identity team handles the **IAM solution** and products and the SecOps team handles the **XDR functionality**. In Microsoft terminology, the identity team looks at **Microsoft Entra ID** (formerly **Azure AD**) and **AD**. SecOps then looks at **Defender XDR** and **Microsoft Sentinel**. Some organizations only use cloud identities in Microsoft Entra ID, and other organizations use hybrid identities with AD and other third-party identity providers (such as **Okta**). The goal of an ITDR solution is to get signals from all those areas, regardless of where the identity resides.

In short, we want the capability to *prevent*, *detect*, and *respond* to identity-related threats. If we start thinking about how attacks start, it is typically through phishing or other social engineering tactics, up to more sophisticated attacks where the IAM infrastructure is targeted to exploit vulnerabilities in that area. If the attacker is successful, this can lead to unauthorized access to sensitive information, data exfiltration, ransomware deployment, and more. IAM's job is to ensure that the right people have the right access to files, systems, apps, and so on to be able to do their jobs without positioning those types of resources at any risk for compromise.

In this book, we will be focusing on AD and MDI as our ITDR product. Other ITDR products from Microsoft will then be, as we learned earlier, Microsoft Entra ID, Microsoft Entra ID Protection, and

Microsoft Defender XDR, and if you are invested in the Microsoft security ecosystem, you will then have your entire ITDR solution in place.

It is highly recommended to not just implement MDI as the only protection for your AD deployment but also explore common entry points to be able to reduce the attack surface. Such *close the gap* exercises or best practices could be implementing tiering, not using excessive privileges (least-privileged framework), using privileged access workstation/secure access workstation, isolating or, even better, decommissioning legacy systems, patching, identifying all of your critical assets, and planning for compromise – yes, planning is key because if and when we get our AD and other systems compromised, we need to be prepared and know how to recover them.

Adding MDI into the mix brings a flare of defense to this stormy sea. As you journey through these pages, you'll discover how MDI stands as a front against identity-based threats, using advanced technology such as machine learning and artificial intelligence to detect, investigate, and stop potential breaches before they escalate.

As we begin, remember that you're not just reading a chapter; you're stepping into a very crucial role in safeguarding the digital identities that are the foundation of your organization. This journey is about providing you with the knowledge to not just navigate the complexities of protecting on-premises identities, but to succeed in this sometimes unpredictable digital landscape.

Modern identity threats and strategic defense frameworks

As we continue our exploration of cybersecurity, it's essential to recognize that the nature of identity threats has evolved significantly. Today's adversaries employ increasingly sophisticated methods to exploit and manipulate identities within organizational networks. Unlike earlier attacks that might have relied solely on brute force or basic phishing techniques, modern identity threats are multi-faceted, leveraging advanced tactics to achieve their objectives.

Modern adversaries target identities using a variety of sophisticated techniques, including the following:

- **Credential theft and reuse** : Attackers obtain user credentials through phishing, malware, or social engineering and reuse them across multiple platforms to gain unauthorized access (think about the **solarwinds123** password that caused the SolarWinds hack)
- **Pass-the-Hash (PtH) attacks** : Exploiting stolen hashed passwords, attackers authenticate as legitimate users without needing the actual plaintext passwords
- **Pass-the-Ticket (PtT) attacks** : Utilizing stolen Kerberos tickets, attackers impersonate users to access resources within the network
- **Golden Ticket attacks** : Crafting counterfeit Kerberos **Ticket Granting Tickets (TGTs)** to gain unrestricted access across the entire domain

- **Silver Ticket attacks** : Forging service tickets to access specific services within the network
- **Kerberoasting** : Extracting service account credentials from Kerberos tickets to crack passwords offline
- **DCShadow and DCSync attacks** : Manipulating domain controller functions to inject malicious changes or extract credentials from AD

These tactics allow attackers to navigate through networks silently, access sensitive information, and escalate their privileges, often remaining undetected for extended periods.

IMPLEMENT A ROBUST PASSWORD POLICY

*Ensure that your organization enforces a password policy that prohibits the use of easily guessable elements, such as the company name, in user passwords. It's common to observe that employees create simple passwords incorporating the organization's name (e.g., **CompanyName123**), making them vulnerable targets for attackers. By restricting such patterns, you significantly reduce the risk of unauthorized access through brute force or social engineering attacks.*

Imagine an attack as a carefully orchestrated heist, where each step brings the attacker closer to their goal. The Cyber Kill Chain serves as the blueprint for this heist, breaking down each stage of an attack so you can anticipate and prevent malicious moves before they reach their target.

The Cyber Kill Chain

Developed by *Lockheed Martin*, the **Cyber Kill Chain** is a model that describes the seven stages of a cyberattack, providing a structured approach to understanding and disrupting adversary actions. By dissecting an attack into distinct phases, defenders can identify opportunities to interfere and halt the progression of threats.

For any cyberattack to be successful, it must go through the following:

1. **Reconnaissance** : Gathering information about the target to identify potential vulnerabilities.
2. **Weaponization** : Creating malware or exploit tools tailored to the identified vulnerabilities.
3. **Delivery** : Transmitting the weaponized payload to the target system (e.g., via phishing emails).
4. **Exploitation** : Triggering the exploit to gain unauthorized access to the system.
5. **Installation** : Installing malware to maintain persistence within the network.
6. **Command and Control (C2)** : Establishing communication channels to remotely control the compromised systems.
7. **Actions on Objectives** : Executing the intended goals, such as data exfiltration, ransomware deployment, or system sabotage.

By breaking any one of these stages, the attack will be prevented. Now that we've mapped out the stages of a cyberattack with the Cyber Kill Chain, let's dive deeper into the tactics and techniques attackers use at each stage. This is where the **MITRE Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) framework** comes into play, offering a more granular view of adversary behaviors.

MITRE ATT&CK framework

The **MITRE ATT&CK** framework serves as a comprehensive guide to understanding the behaviors and methods of cyber adversaries. It's like a master playbook that maps out how attackers operate, based on real-world observations. The framework breaks down the phases of an attack, offering a detailed taxonomy that helps security teams anticipate threats, enhance their defenses, and respond more effectively.

Core components of the MITRE ATT&CK framework include the following:

- **Tactics** : These represent the adversary's high-level objectives or goals, such as gaining initial access or escalating privileges within a network. They provide insight into why certain actions are taken by attackers during an intrusion.
- **Techniques** : These are the specific methods adversaries use to achieve their tactical goals. For instance, attackers might use phishing emails for initial access or credential dumping to gain account details.
- **Sub-techniques** : These dive deeper, offering more granular methods that attackers might employ. For example, under credential dumping, different sub-techniques detail how credentials might be extracted using various tools or methods.

The Enterprise Matrix of MITRE ATT&CK encompasses 14 tactics, each representing a critical phase in a cyberattack:

1. **Reconnaissance** : Identifying potential targets and gathering information.
2. **Resource Development** : Building the capabilities and infrastructure needed for an attack.
3. **Initial Access** : Breaking into the target network using methods such as phishing or exploiting vulnerabilities.
4. **Execution** : Running attacker-controlled code on a compromised system.
5. **Persistence** : Maintaining access to the environment over time through methods such as backdoors or configuration changes.
6. **Privilege Escalation** : Gaining higher access levels to penetrate deeper into the network.
7. **Defense Evasion** : Implementing techniques to avoid detection or circumvent security measures.
8. **Credential Access** : Stealing usernames, passwords, and other credentials.
9. **Discovery** : Mapping out the network environment to find valuable targets.
10. **Lateral Movement** : Spreading access to other systems in the network.
11. **Collection** : Gathering data of interest before exfiltrating it.
12. **Command and Control (C2)** : Establishing a communication link with compromised systems.
13. **Exfiltration** : Moving stolen data out of the organization.
14. **Impact** : Disrupting or manipulating systems and data to fulfill the attacker's objectives, such as deploying ransomware or deleting data.

To illustrate how the MITRE ATT&CK framework can be applied to real-world scenarios, let's look at an example of an attack targeting AD. This example focuses on a common method attackers use to extract sensitive credentials, known as a DCSync attack. By breaking down the tactic, technique, and

sub-technique involved, we can see how MDI plays a crucial role in detecting and mitigating such threats before they can escalate. Here's how this type of attack typically unfolds:

- **Tactic – Credential Access :**

Credential access involves techniques attackers use to obtain credentials, such as usernames, passwords, or token hashes, from a system or network, allowing them to access additional resources.

- **Technique – T1003 – OS Credential Dumping :**

This technique involves extracting credential data from operating systems. It can include direct access to credentials stored in memory or on disk, such as hashes, passwords, or tickets.

- **Sub-technique – T1003.006 – DCSync :**

A DCSync attack allows an attacker to simulate the behavior of a domain controller and request user credential data from other domain controllers. By abusing replication permissions, the attacker can extract credentials from the AD database directly.

An attacker who has gained elevated permissions, such as *Replicating Directory Changes* or *Replicating Directory Changes All* rights, can execute a **DCSync attack** using tools such as **Mimikatz**. By impersonating a domain controller, the attacker can request password hashes for accounts, including `krbtgt` (the Kerberos Ticket Granting Ticket (TGT) account) or highly privileged domain admins.

Once the attacker obtains the password hashes, they can leverage them in other attacks, such as Pass-the-Hash (PtH) or Golden Ticket attacks, enabling them to move laterally across the network or maintain persistent access with administrative privileges.

The MITRE ATT&CK framework is more than just a list of tactics; it's a way to visualize and understand the full scope of an attack. When organizations map incidents to this framework, they gain valuable insights into attacker behavior, helping them predict potential future moves and adjust their defenses accordingly. This structured approach allows security teams to prioritize their response and plug gaps that could be exploited by adversaries.

While the MITRE ATT&CK framework provides an extensive catalog of attacker tactics and techniques, understanding the broader sequence and integration of these actions is crucial. This leads us to the **Unified Kill Chain (UKC)**, a comprehensive model that combines the strengths of both the Cyber Kill Chain and MITRE ATT&CK to provide an 18-stage approach for tackling advanced cyber threats.

The Unified Kill Chain

The **UKC** is an advanced model that builds upon the traditional Cyber Kill Chain by incorporating additional dimensions to provide a more comprehensive and integrated approach to threat detection

and response. Developed by *Paul Pols*, the UKC includes 18 distinct phases that detail the tactics used by **Advanced Persistent Threats (APTs)** and ransomware groups. Unlike the traditional kill chain, which is primarily perimeter and malware-focused, the UKC addresses a broader range of attack vectors and includes socio-technical elements, such as social engineering and pivoting.

The phases of the UKC include the following:

1. **Reconnaissance** : Conducting thorough research to identify and select targets using both active and passive information-gathering techniques.
2. **Resource Development** : Engaging in preparatory activities to establish the necessary infrastructure, such as registering domains or developing malware, required for executing the attack.
3. **Delivery** : Transmitting the malicious payload or weaponized object into the target environment through various methods, such as phishing emails, malicious attachments, or drive-by downloads.
4. **Social Engineering** : Manipulating individuals within the organization to perform actions that compromise security, such as divulging credentials or unwittingly executing malicious code.
5. **Exploitation** : Taking advantage of identified vulnerabilities within systems to execute malicious code, thereby gaining unauthorized access.
6. **Persistence** : Establishing a long-term presence within the compromised system to ensure continued access, often by creating backdoors or modifying system configurations.
7. **Defense Evasion** : Implementing techniques to avoid detection by security tools and personnel, such as obfuscating code or disabling security features.
8. **Command and Control (C2)** : Setting up communication channels to remotely control the compromised systems, enabling the attacker to issue commands and receive data.
9. **Pivoting** : Using the compromised system as a launch point to access and infiltrate additional systems within the network that were previously inaccessible.
10. **Discovery** : Gathering detailed information about the network environment, including system configurations, network topology, and security measures, to identify further targets.
11. **Privilege Escalation** : Increasing the level of access within the network by exploiting vulnerabilities or misconfigurations, allowing deeper penetration into the organization's systems.
12. **Execution** : Running attacker-controlled code on local or remote systems to perform malicious activities, such as data manipulation or further exploitation.
13. **Credential Access** : Stealing or obtaining access to system, service, or domain credentials to facilitate ongoing unauthorized access and movement within the network.
14. **Lateral Movement** : Traversing the network horizontally to access additional systems and resources, expanding the attacker's reach within the organization.
15. **Collection** : Identifying and aggregating valuable data from various sources within the network in preparation for exfiltration or misuse.
16. **Exfiltration** : Transferring the collected data out of the target network to an external location controlled by the attacker, often using encrypted channels to avoid detection.
17. **Impact** : Executing actions that disrupt, manipulate, or destroy systems and data, such as deploying ransomware or altering critical information.

18. **Objectives** : Achieving the overarching goals of the attack, which may include financial gain, data theft, espionage, or sabotage, aligning the tactical actions with strategic outcomes.

To further explain the UKC, it breaks down cyberattacks into three overarching phases: In, Through, and Out. Understanding these phases provides a clearer roadmap for anticipating attacker movements and deploying targeted defenses:

- **In** : This initial phase encompasses all the steps attackers take to infiltrate a target network. It includes activities such as reconnaissance, resource development, payload delivery, social engineering, exploitation of vulnerabilities, and establishing persistence. Essentially, this is where attackers gain their foothold and set up their operations within the network.
- **Through** : Once inside, attackers move deeper into the network, escalating their privileges and expanding their access. This phase involves pivoting, discovery, privilege escalation, execution, credential access, and lateral movement across the network. Think of this as attackers navigating through various layers of the network to reach more valuable targets.
- **Out** : In the final phase, attackers execute their primary objectives, which could range from data exfiltration and ransomware deployment to system sabotage. This stage includes activities such as data collection, exfiltration, and causing impact on the target's systems or data integrity. It's the finale of the attack, where the attackers achieve their goals and attempt to cover their tracks or escape the network.

The real strength of the UKC is its flexibility. It recognizes that attackers don't always follow a set path – they might skip steps or switch strategies as they go. This means defenders need to be ready to adapt, using layered security measures that can disrupt attackers at multiple points.

By including phases such as *social engineering* and *pivoting*, the UKC emphasizes the human element in cyber threats. Attackers often manipulate people to gain access, so training employees to recognize these tactics is as important as deploying technical defenses.

The UKC also helps organizations improve both proactive and reactive responses. It guides where to focus early defenses, such as detecting reconnaissance, and how to respond if an attacker gains access, such as by containing lateral movement.

READ MORE

For a deeper dive into the UKC and its 18 phases, visit <https://www.unifiedkillchain.com>. This resource provides detailed explanations, examples, and additional insights.

Now that we've looked at the different stages of a cyberattack, it's time to explore how tools such as MDI fit into those frameworks.

MDI's strategic position in the cybersecurity ecosystem

MDI is designed to protect on-premises AD identities, our core foundation of the organization, and its primary feature is to monitor user activities, network traffic, Windows events, and entity behaviors continuously. It uses advanced algorithms and machine learning to detect unusual activities, such as

lateral movement, privilege escalation, and reconnaissance attacks, which could indicate potential security breaches.

Today, it's more common that we still use AD as our **Source of Authority (SoA)** and synchronize the identities to Microsoft Entra ID. It's also still common that we are using **Active Directory Federation Services (AD FS)** and **Active Directory Certificate Services (AD CS)**. These workloads must be protected at all times. Just because we are still required to keep using those legacy systems, they are critical workloads and should be treated as Tier 0 systems a.k.a. your most critical asset. We will not go into detail about how to protect your Tier 0 systems with the Enterprise Access Model or the legacy AD tier model in this book, but it is highly recommended to read and learn about those control strategies.

EXPLORE MORE ABOUT ENTERPRISE ACCESS MODEL

Learn more about the Enterprise Access Model and legacy AD tier model on Microsoft Learn: <https://learn.microsoft.com/en-us/security/privileged-access-workstations/privileged-access-access-model>

With this in mind and our identity landscape in place, we need to start getting signals from these components because all parts of the landscape can be attacked. In this modern world, we want to gather our signals from on-premises (from AD DS, AD CS, AD FS, and Entra Connect) together with cloud signals, such as Microsoft Entra ID, Defender for Cloud Apps, Defender for Endpoint, and so on, to be able to correlate and analyze the data. The more signals we have, the more the likelihood of detecting anomalies and seeing the full picture or story of an attack increases.

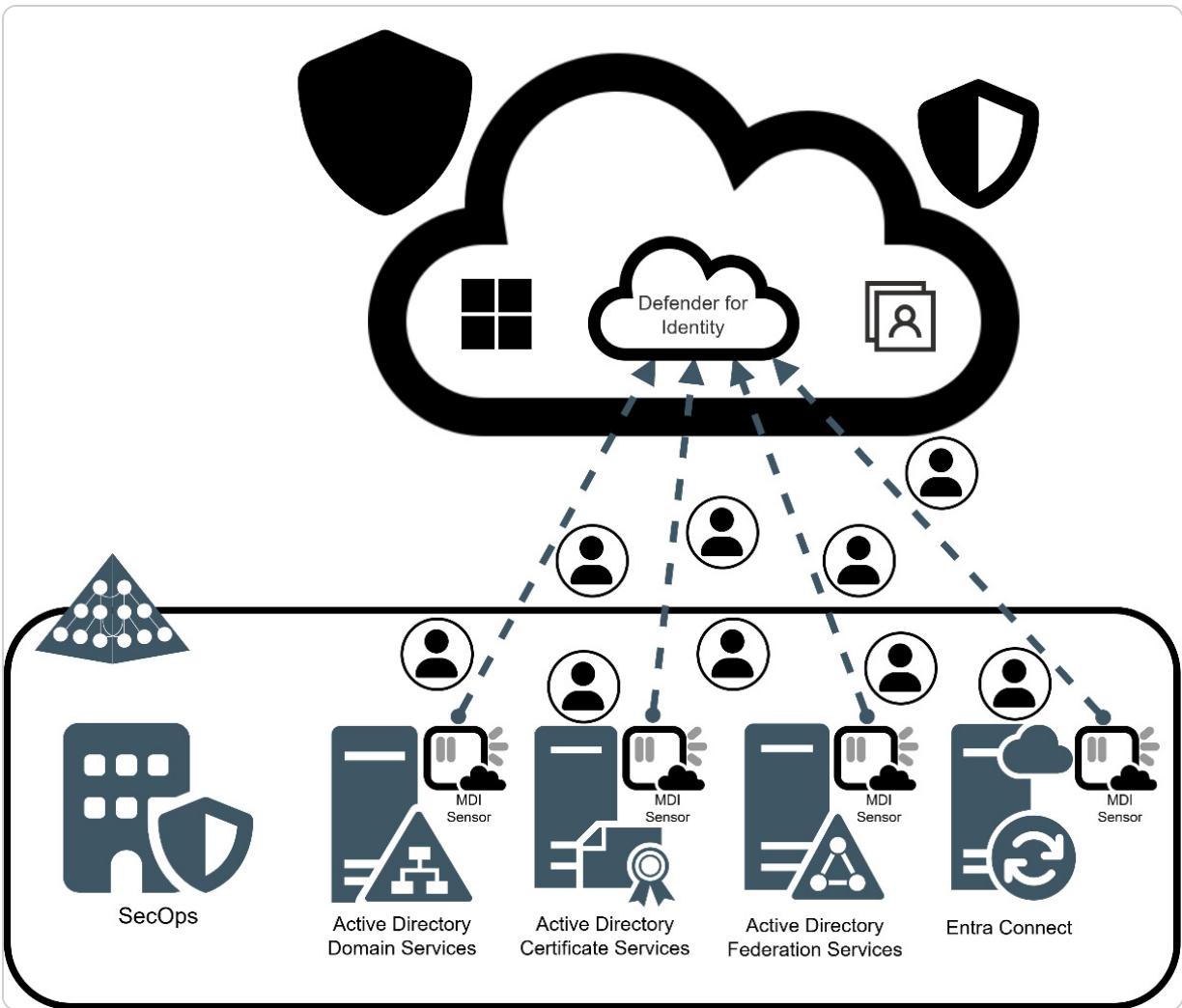


Figure 1.1 – MDI high-level architecture

The preceding figure shows the importance of integrating personas and departments, and as the journey in this book continues, we will see exactly how important it will be. We will be informed about the security posture, if we have misconfigurations in the identity infrastructure, getting detected about privileged accounts, service accounts, and privileged escalation paths.

One of the absolute most time-consuming tasks for SOC (Security Operation Center) analysts is *user profiling*. The concept of criminal profiling, which special agent John Edward Douglas invented in the late 1970s at the United States *Federal Bureau of Investigation (FBI)* as part of the Behavioral Science Unit, laid the groundwork for modern criminal profiling, involving in-depth interviews with criminals to understand their psychopathologies and behavioral patterns. Today, in the world of cyber and cybersecurity, we use something called **User Entity Behavior Analytics (UEBA)**, which will create a baseline of the user and their behavior, and when a user steps out of that baseline and the behavior, the UEBA will detect that anomaly and alert for that security risk. If you have activated UEBA in

Microsoft Sentinel, you can look at the UEBA table, `BehaviorAnalytics`, to see if you have any data ingested. Here's the book's first KQL query:

```
BehaviorAnalytics  
| take 20
```

In this KQL query, you will look at the latest 20 entries of the `BehaviorAnalytics` table.

For MDI, we want to have the capability to detect identity-specific threats, such as reconnaissance and lateral movement, including PtH, Golden Ticket, PtT, password spray, and attacks against the identity infrastructure. These types of advanced threats are part of what we call the cyberattack kill chain. This means they include unusual actions or behaviors that occur at any stage of a cyberattack, from initial reconnaissance to the final execution of the attack. This concept helps in identifying and understanding potential threats by analyzing activities at each phase of an attack, enabling timely detection and response actions.

Unpacking key features and benefits of MDI

MDI is designed to protect against identity-based attacks in hybrid environments, providing a vigilant watch over AD and Microsoft Entra ID. MDI continuously monitors user activities and authentication patterns, offering real-time alerts and insights to detect threats such as credential theft, lateral movement, and privilege escalation.

These are the key features of MDI:

- **Behavioral analytics**: MDI builds a baseline of normal user behaviors to spot unusual activities, such as unexpected logins or data access. This helps catch compromised accounts early before attackers can cause serious harm.
- **Advanced threat detection**: MDI excels at detecting sophisticated attack techniques, such as PtH, PtT, Golden Ticket, Silver Ticket, and Kerberoasting. It also catches threats such as DCShadow and DCSync, providing insights into potential AD manipulation.
- **Lateral Movement Path (LMP) analysis**: MDI maps potential pathways attackers could use to access more valuable resources within the network. By visualizing these paths, security teams can proactively secure critical assets, making it harder for attackers to gain a foothold.
- **Enhanced monitoring of AD CS, AD FS, and Entra Connect**: Recent updates improve MDI's visibility into AD CS, AD FS, and Microsoft Entra Connect. This allows MDI to detect suspicious activities in federated identity setups and certificate-based authentications, while also ensuring secure identity synchronization between on-premises AD and Entra ID.
- **Integration with Microsoft Sentinel and Defender XDR**: MDI connects seamlessly with the Defender XDR portal, enabling advanced hunting and centralized management of identity-related incidents. This integration allows analysts to correlate identity threats with data from endpoints and cloud resources, creating a unified view of security incidents.
- **Proactive attack disruption**: MDI is a key part of Defender XDR's automatic attack disruption capabilities. It can automatically trigger actions such as disabling suspicious user accounts, helping to contain threats quickly and prevent further damage.

Identity is often the initial target for attackers. MDI provides organizations with the tools needed to detect, investigate, and respond to these threats quickly. Its close integration with other Microsoft security tools ensures that identity-related insights are part of a larger defense strategy, allowing security teams to stay a step ahead of adversaries. With improved visibility into AD CS, AD FS, and Entra Connect, MDI ensures that hybrid identity flows remain secure and monitored.

Summary

In this chapter, we delved into the growing identity threat landscape and the vital role that MDI plays in addressing these challenges. We explored the modern identity-based threats organizations face, the defense frameworks that help mitigate these risks, and how MDI strategically fits into the broader cybersecurity ecosystem. We also highlighted the key features of MDI, showing how it strengthens defenses against advanced attacks, providing critical ITDR capabilities.

As we move on to the next chapter, the focus shifts to practical deployment. I'll guide you through important pre-installation planning to ensure you're prepared, followed by a detailed step-by-step guide for deploying MDI. You'll also learn how to configure proxy settings and ensure a successful setup by conducting vital post-installation activities.

2

Setting up Microsoft Defender for Identity

In this chapter, we'll lay the foundation for implementing **Microsoft Defender for Identity (MDI)**, setting the stage for a successful deployment. We'll begin by ensuring your environment is properly prepared, covering everything you need to know before starting the deployment. From there, you'll find a detailed walk-through of the installation process, making sure you have a smooth experience from start to finish.

For those with unique network needs, we'll also delve into how to configure your environment for secure and efficient communication. To wrap things up, we'll focus on critical post-deployment checks, helping you validate that MDI is working as intended and ready to secure your Active Directory environment.

I do think these skills and knowledge are vital for IT professionals and cybersecurity professionals who are tasked with safeguarding their organization from getting Active Directory compromised, because let's be real – Active Directory is a highly attractive target for adversaries: a successful breach can lead to unauthorized access or, even worse, adversaries getting full control of the entire environment. The result would be a great financial loss.

By the end of this chapter, you'll have a clear roadmap for implementing MDI, from initial setup to post-installation checks.

In this chapter, we will cover the following:

- Pre-installation and planning checklist: laying the groundwork
- Deployment of MDI- a step-by-step guide
- Navigating step-by-step proxy configuration for MDI
- Ensuring success with post-installation activities

Let's get started!

NEWS FROM MICROSOFT IGNITE 2024: UNIFIED AGENT

Microsoft has introduced a unified agent that integrates Microsoft Defender for Endpoint (MDE) with Microsoft Defender for Identity (MDI), extending protection across endpoints, operational technology (OT) devices, identities, and Data Loss Prevention (DLP). This consolidation simplifies deployment and maintenance by eliminating the need for separate agents, thereby reducing system overhead and enhancing efficiency. Organizations can now enable MDI directly from the Defender portal, streamlining the process of securing on-premises identities.

Technical requirements

Given the expansive nature of Microsoft 365 and its associated ecosystem, setting up and optimizing MDI demands specific prerequisites. These requirements are essential for ensuring seamless integration and functionality within the Microsoft 365 framework, catering to both security needs and operational efficiency.

You will require the following:

- **A Microsoft tenant**
- A Microsoft subscription that includes **Microsoft Defender for Identity**, such as **Microsoft 365 E5** or **Microsoft 365 E3 + E5 Security**
- Basic **Microsoft 365** knowledge
- Active Directory knowledge
- A **virtual or physical server environment** with **Active Directory** installed and configured:
 - **Optional** : Active Directory Federation Services and Active Directory Certificate Services installed and configured
- Basic **PowerShell** knowledge
- Basic networking knowledge
- **Optional** : If you want to follow along with proxy setup and configuration, you need one or two virtual machines with **Ubuntu 22.04** or later installed:
 - Make the installation of Ubuntu a minimal server installation

All the code examples for this chapter can be found on GitHub at

<https://github.com/PacktPublishing/Microsoft-Defender-for-Identity-in-Depth/tree/main/Chapter02>

Pre-installation and planning checklist: laying the groundwork

It's wonderful that we are now in the planning phase. But before installing MDI, it's crucial to prepare a thorough checklist. This preparation ensures that the deployment process is smooth, and that MDI integrates seamlessly into your environment. By taking the time to properly plan, you can avoid common challenges and ensure that your organization is ready to leverage MDI's capabilities to their fullest potential:

1. You need to understand the licensing requirements, understand that current service doesn't allow for limiting features to specific users, understand that MDI is a tenant-level activation, and familiarize yourself with the types of data MDI collects to effectively profile user behavior.
2. Additionally, consider Microsoft's strategies for mitigating malicious insider activities, particularly regarding high-privilege roles.
3. Lastly, assess your infrastructure to meet the sizing recommendations, ensuring MDI operates efficiently within your environment.

AZURE ADVANCED THREAT PROTECTION (ATP) OR MICROSOFT DEFENDER FOR IDENTITY

Please be aware that MDI, formerly known as Azure ATP, is part of a broader rebranding by Microsoft to align its security products under the Microsoft Defender suite. This change reflects the product's enhanced focus on identity security within the Microsoft security ecosystem. You will see that the old name is still there in some configurations and some portals.

Let's begin by exploring all the prerequisites needed for a successful MDI deployment. We'll cover several important steps in this planning phase to ensure a successful deployment by the end of this chapter.

First, we will dive into licensing – yes, it is fun and cool, but perhaps not your favorite. After that, we will discuss the required permissions, followed by the operating system and server requirements. We will then move on to networking, more specifically the ports and URLs that need to be opened.

Following that, I will guide you through installing the MDI PowerShell module and explain the types of Windows events that MDI needs to be fully functional. We will also cover how user profiling works, sizing requirements, and the necessary steps for installing the MDI sensor on **Active Directory Federation Services (AD FS)**, **Active Directory Certificate Services (AD CS)**, and Entra Connect server. Lastly, we will dive into how service accounts work for MDI.

Licensing

First on the list is licensing: we need one of the following licenses activated in our tenant:

- Enterprise Mobility + Security E5 (EMS E5/A5)
- Microsoft 365 E5 (Microsoft E5/A5/G5)
- Microsoft 365 E5/A5/G5 Security
- A standalone **Defender for Identity** license

NOTE

You know how fast Microsoft changes things, so check the MDI licensing requirements at learn.microsoft.com.

As of November 2023, the **Service Level Agreement (SLA)** for Microsoft Online Services, which MDI is part of, states that you can have some credits from the licensing bill if the uptime (in percentage) drops below 99.9%.

The definition of *downtime* is any period of time where the administrator is unable to access the MDI portal.

The uptime percentage is calculated using the following formula:

$$\frac{\text{User Minutes} - \text{Downtime}}{\text{User Minutes}} \times 100$$

The credit you can get is as follows:

Uptime Percentage	Credit
< 99.9%	10%
< 99%	25%

Table 2.1 – SLA credit

A question a lot of administrators and companies ask is whether they need to license all users in their tenant if they want to utilize MDI, and the answer is **yes**. MDI is a tenant-wide activation, and like other tenant services, it is not currently capable of limiting benefits to specific users.

What permissions do you need?

In Microsoft Entra ID, you will need to have at least the role of **Security Administrator**. When you first enter the Microsoft Defender portal and go to the **Settings** blade, you need to have the appropriate permissions; otherwise, your MDI workspace will not be created. You will learn more about **Role-Based Access Control (RBAC)** later in this chapter.

What are the operating system requirements?

Before you read the following list, make sure that you are running a version of Windows Server that is supported by Microsoft and that you don't need to take care of it in a couple of months.

The supported Windows Server editions, which includes the Server Core and Desktop Experience installations, are as follows:

- Windows Server 2016
- Windows Server 2019
- Windows Server 2022

See the following table for the support end dates:

Operating System Version	Start Date	Mainstream Support End Date	Extended Support End Date
Windows Server 2016	October 15, 2016	January 11, 2022	January 11, 2027

Windows Server 2019	November 13, 2018	January 9, 2024	January 9, 2029
Windows Server 2022	August 18, 2021	October 13, 2026	October 14, 2031

Table 2.2 – Windows Server OS versions and support dates

Extended Support is offered five years after mainstream support, and during this time, Microsoft will provide security updates, bug fixes, and reliability updates.

Do not confuse the Extended Support with **Extended Security Updates (ESU)**, which is a last-resort option for customers who need to run legacy Microsoft products. ESU only includes Critical and/or Important security updates, which are defined by the **Microsoft Security Response Center (MSRC)**. ESU will be offered for a maximum of three years after the Extended Support end date.

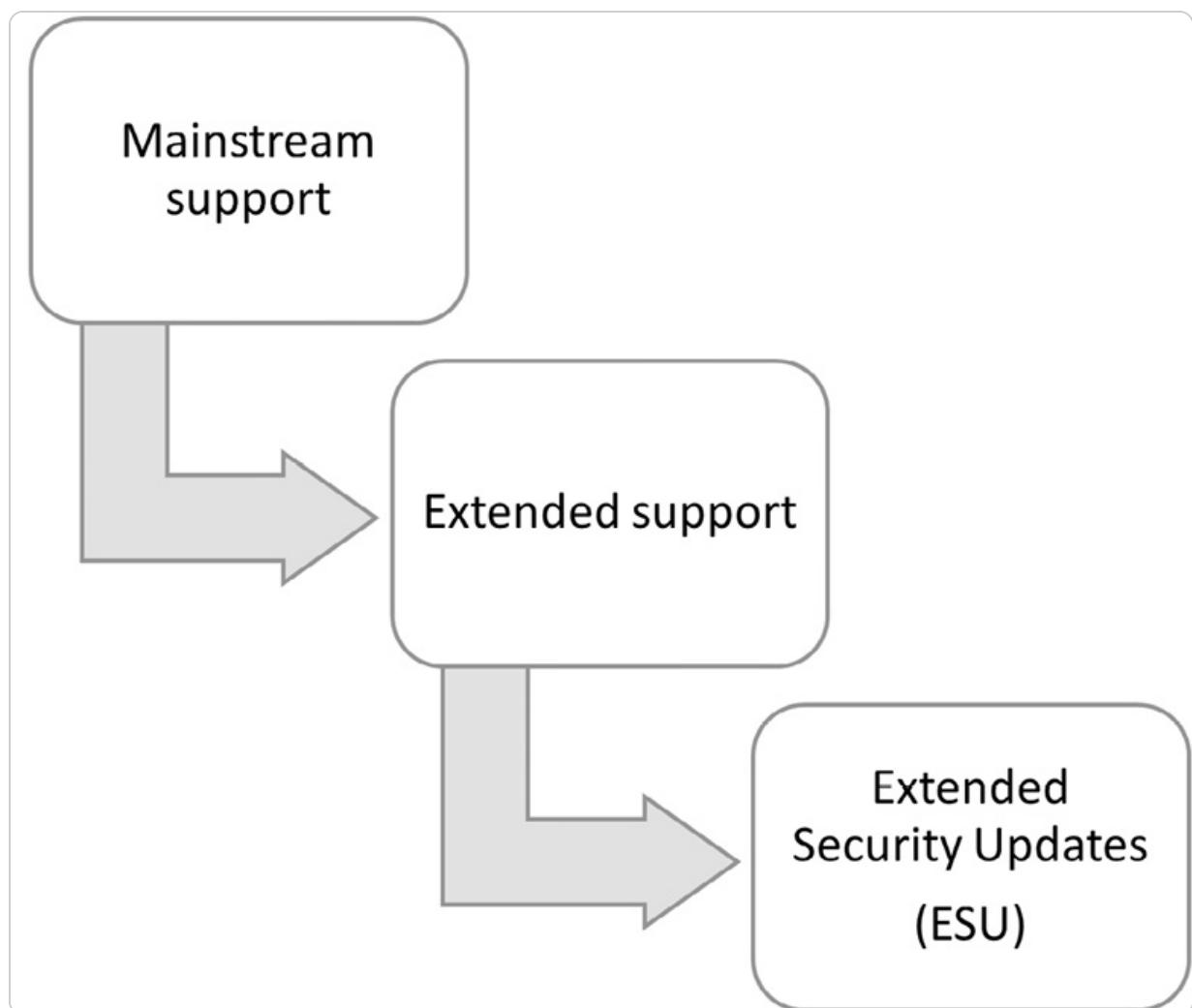


Figure 2.1 – Overview of Microsoft Support phases

Currently, you can buy ESU licenses through Azure for your Windows Server 2012/2012 R2, and if you are running the MDI sensor on that version of Windows Server, it's highly recommended to upgrade to a newer version of Windows Server. The MDI sensor will be operational for a short period of time, and can even get updates, but some functionalities require a newer operating system version.

So, if you are installing new domain controllers, or other Tier 0 systems, and other types of servers that the MDI sensor supports, it's highly recommended to (as of today) install Windows Server 2022.

Other sensor requirements

Now that you have sorted out the operating system version, I bet you want to know how you should specify the server. The following table outlines the necessary server requirements for MDI to run effectively.

Type	Requirements
Cores	Minimum of two cores.
RAM	Minimum of 6 GB. The following requirements apply when the MDI server is installed as a virtual machine. We allocate all memory to ensure performance and stability, as dynamic memory allocation can lead to resource competition and unpredictable behavior, which may affect the MDI sensor's efficiency and reliability: <ul style="list-style-type: none">• For Hyper-V : Do not enable Enable Dynamic Memory• For VMware : Make sure to select the Reserve all guest memory (All locked) option in the VM settings
Disk	Recommended to have at least 10 GB free for MDI logs and binaries.
Performance	Set the power option to High Performance .
Pending reboot, or reboot required after MDI installation	For a smooth MDI installation, plan a maintenance window due to potential reboot, especially if a pending reboot exists or .NET Framework 4.7 installation is needed. If .NET Framework 4.7 or newer isn't pre-installed, MDI setup will initiate its installation, possibly requiring a system reboot for completion.

Table 2.3 – Server requirements for MDI

The MDI sensor actively measures and manages the compute and memory capabilities on its host server, ensuring optimal functionality without obstructing the server's primary operations. By conducting assessments every 10 seconds, the MDI sensor dynamically adjusts its usage to maintain a minimum of 15% free resources. In scenarios where the server's resources are heavily overloaded, the sensor may limit its monitoring scope, which could lead to an MDI sensor health alert indicating **Dropped port mirrored network traffic** on the MDI sensor page.

Networking

The MDI sensor must have a clear view of sight for its communication with the Defender for Identity cloud service, and we have some options to make that happen. We are going from the hardest option to the easiest one, but also from the most private option to the most public option:

- **Azure ExpressRoute** allows for the integration of your local on-premises networks into the Microsoft cloud over a private connection with the help of a connectivity provider, such as a telco. If you are choosing this option, you must add MDI service BGP community (12076:5220) into the route filter. Be aware that you need the ExpressRoute of Microsoft peering.
- **Forward proxy** allows you to connect your MDI sensors to the MDI cloud service, some configuration is needed, and we will go through how to configure the MDI sensor and which URLs that need to be in the allow list.
- **Firewall** is the third and last option. Network administrators need to open IP addresses, or URL (if supported) to the MDI cloud service. Some firewall vendors have the option to subscribe to the *Azure IP Ranges and Service Tags* file. If you have your own automation for this, please use the *AzureAdvancedThreatProtection* service tag.

IMPORTANT NOTE

SSL inspection and interception are incompatible because they disrupt the authentication process.

As may have experienced with Microsoft Cloud services, or are about to experience, a well-configured network is so important. Why? Because without it, MDI cannot alert you when it detects something malicious. Similarly, with **Microsoft Defender for Endpoint (MDE)**, you wouldn't be able to isolate or perform live response on infected machines. Therefore, our devices need to communicate properly with all necessary components. In the following subsections, we will outline the specific ports and URLs that need to be configured to allow this seamless operation and communication with Microsoft's backend.

Ports

Only one port needs to be open to the internet in the firewall, which is the **outbound TCP port 443**. This port is necessary for secure HTTPS communication initiated from the MDI sensors within the network to the MDI service, accessible through `*.atp.azure.com`.

For Internal communication to and from the MDI sensor to devices on the network, we need the following:

Protocol	Transport and Port	From	To
DNS	TCP/53 and UDP/53	MDI sensor	All DNS servers
Netlogon (SMB, CIFS, SAM-R)	TCP/445 and UDP/445	MDI sensor	Every device connected to the network
RADIUS	UDP/1813	RADIUS	MDI sensor
NTLM over RPC	TCP/135	MDI sensor	Every device connected to the network
NetBIOS	UDP/137	MDI sensor	Every device connected to the network

Table 2.4 – Internal communication requirements for the MDI sensor

If you are working with multiple forests in Active Directory, you need to look at the following list and have these ports opened, outbound, to get successful communication:

Protocol	Transport and Port	From/To
LDAP	TCP/389 and UDP/389	Domain controllers
Secure LDAP	TCP/636	Domain controllers
LDAP to Global Catalog	TCP/3268	Domain controllers
LDAPS to Global Catalog	TCP/3269	Domain controllers

Table 2.5 – Port requirements for multi-forest Active Directory

URLs

Here is a list of URLs for the MDI cloud service. This list is especially useful if you are using a forward proxy or need to verify the URLs in your firewall.

These are the URLs for the proxy or firewall explicit allowlists:

- < your-workspace-name>sensorapi.atp.azure.com
- crl.microsoft.com
- ctld1.windowsupdate.com
- www.microsoft.com/pkiops/*
- www.microsoft.com/pki/*

To find your workspace name for MDI, you need to go to the Defender portal or to this URL:
<https://security.microsoft.com/settings/identities?tabid=about> .

You can, with the `Invoke-WebRequest` PowerShell cmdlet, see if you have connectivity to your MDI workspace. So, for the fictitious company Contoso Bank, the workspace name could be `contosobank` :

```
Invoke-WebRequest https://<your-workspace-
name>sensorapi.atp.azure.com/tri/sensor/api/ping
Invoke-WebRequest https://contosobanksensorapi.atp.azure.com/tri/sensor/api/ping
```

Here it is if using `curl` :

```
curl https://<your-workspace-name>sensorapi.atp.azure.com/tri/sensor/api/ping
```

The next section will talk about the MDI PowerShell module; if you follow the steps in that section, you can come back here later to do the connectivity test to the MDI service:

```
Test-MDISensorApiConnection -BypassConfiguration -SensorApiUrl 'https://<your-
workspace-name>sensorapi.atp.azure.com' -ProxyUrl
'https://proxy.contosobank.com:8080'
```

PowerShell

It's essential to install the **MDI PowerShell module**, as it contains important functions that help in configuring and validating the MDI environment for optimal operation.

```
Install-Module -Name DefenderForIdentity
```

LEARN MORE

Get more insights into the MDI PowerShell module in the PowerShell Gallery:

<https://www.powershellgallery.com/packages/DefenderForIdentity>.

See [Chapter 3 , Leveraging MDI PowerShell for Automation and Management](#) , for information on the manual installation of the module if you aren't allowed to install this module on servers where the MDI sensor will be installed.

It's highly recommended to get the current state of the environment with the following cmdlet:

New-MDIConfigurationReport

Before diving into its usage, it's important to understand the syntax and how each parameter functions. The following is the syntax format for this cmdlet, which outlines the structure and available options:

```
New-MDIConfigurationReport [-Path] <String> [-Mode] <String> [-OpenHtmlReport]
```

- **Path** determines where reports will be saved
- **Mode** allows for selection between Domain and LocalMachine modes; the former sources settings from Group Policy objects while the latter does so from the local machine
- **OpenHtmlReport** triggers the HTML report to open once it's created

Here's an example:

```
New-MDIConfigurationReport -Path "C:\Reports" -Mode Domain -OpenHtmlReport
```

Another script that performs readiness checks on domain controllers and certificate servers is `Test-MdiReadiness.ps1`. You can find this script in the official MDI GitHub repository:

<https://github.com/microsoft/Microsoft-Defender-for-Identity/tree/main/Test-MdiReadiness>.

However, for this book, we will proceed with the new official MDI PowerShell module because it offers improved functionality, seamless integration, and is regularly updated by Microsoft. The module provides a more streamlined and efficient way to perform readiness checks compared to the standalone script.

This script will assess the domain controllers for the following:

- Advanced audit policy configuration
- NTLM auditing
- The power scheme being set to high performance
- Root certificates being up to date

For certificate servers, the script will assess the CA servers for the following items:

- Advanced audit policy configuration for CA servers
- CA auditing
- The power scheme being set to high performance
- Root certificates being up to date

After you have run the script, you will get an HTML report and a `.json` file with the collected result for further action, if necessary.

Data collection

The MDI service carefully gathers and preserves data from your designated servers, such as domain controllers, servers that have the MDI sensor (more on that later), and member servers, utilizing this data for comprehensive administrative, tracking, and reporting objectives. The range of collected data is broad, encompassing network interactions, security logs from Windows events, detailed Active Directory configurations, and specific entity details, including personal identifiers (such as names, email addresses, and phone numbers). This extensive collection is fundamental to Microsoft's proactive stance when it comes to detecting attack indicators and generating timely alerts, equipping your SecOps team with deep insights into threat-related activities within your network. Microsoft is committed to using this data exclusively for enhancing the service, ensuring a clear boundary against any other usage, particularly for advertising purposes.

The collected information includes the following:

- Active Directory information (sites, subnets, structure)
- Security logs (Windows Security events)
- Entity data (names, email addresses, phone numbers)
- Network traffic logs, to and from domain controllers (Kerberos authentication, DNS queries, NTLM authentication)

The general Windows events that are required for the MDI sensor are as follows:

- **4662** : An operation was performed on an object
- **4726** : User account deleted
- **4728** : Member added to global security group
- **4729** : Member removed from global security group
- **4730** : Global security group deleted
- **4732** : Member added to local security group
- **4733** : Member Removed from local security group
- **4741** : Computer account added
- **4743** : Computer account deleted
- **4753** : Global distribution group deleted
- **4756** : Member added to universal security group
- **4757** : Member removed from universal security group
- **4758** : Universal security group deleted
- **4763** : Universal distribution group deleted
- **4776** : Domain controller attempted to validate credentials for an account (NTLM)
- **5136** : A directory service object was modified
- **7045** : New service installed
- **8004** : NTLM authentication

(See the official list of Windows events at Microsoft Learn: <https://learn.microsoft.com/en-us/defender-for-identity/deploy/event-collection-overview#other-required-windows-events>.)

To collect these types of events, it's necessary to configure audit policy settings, NTLM auditing, and domain object auditing. The previous list can be mapped to audit policies; for example, the event **4741 - Computer Account Added** is found in **Advanced Audit Policy Configuration > Account Management > Audit Computer Account Management**.

We will configure all the necessary settings in [Chapter 3](#) with the new Defender for Identity PowerShell module.

User profiling

As you just read in the previous section, MDI collects and analyzes the network traffic using deep packet inspection, and in addition to that, MDI also collects some Windows Security events to create a profiling of all users. MDI does not only collect or monitor users, but it also looks at other types of accounts, such as computer accounts, and they can be both domain-joined and not domain-joined. The devices can also be non-Windows and include mobile devices. Essentially, MDI monitors anything that attempts to authenticate or make authorization requests against Active Directory.

MDI alerts use learning periods to understand normal patterns and spot unusual ones. Each alert has its own rules to tell apart normal from suspicious behavior, including set levels of sensitivity and checks for common actions.

We have the option to activate a **test mode** for some of the alerts and detections. This is recommended to activate when we want to learn and understand the MDI alerts for a successful evaluation.

On the **Adjust alert thresholds** page, you can change how sensitive certain alerts are to control how many you get. If you're doing lots of tests, you might want to lower these levels to receive more alerts for understanding purposes. Alerts will pop up right away if you choose **Recommended test mode** or set levels to **Low** or **Medium**, even if the learning period is completed.

There are different modes for the alert threshold:

- The **Low** and **Medium** thresholds generate more alerts
- The **High** threshold is the default value and reduces false positives

To get more context and understanding about the alerts, we need to understand that the learning period varies, ranging from there being no learning period at all to it being several weeks. Some types of alerts won't be visible during the learning period.

For example, If we look at the alert **Security principal reconnaissance**, MDI needs a learning period of 15 days per computer to profile and learn about legitimate users, and for the first 10 days, no alerts

will be triggered.

Sizing

When we looked at the requirements for the server where we want to install the MDI sensor, it's important that, in this case, the domain controllers have the necessary compute and memory resources for the sensor to be operational. If the MDI sensor doesn't have the necessary resources, health alerts will be generated.

Note that the sizing of resources is important for domain controllers and *not* AD CS, AD FS, or Entra Connect servers..

Microsoft previously provided a tool called the **MDI Sizing Tool** to assist with capacity planning for MDI. However, this tool hasn't been updated to support Windows Server 2016 and later versions, which means it might not offer accurate sizing recommendations for organizations using newer server operating systems. As shown in *Table 2.2*, the preferred operating system at the time of writing is Windows Server 2022, offering extended support until October 14, 2031.

LEARN MORE ABOUT THE MDI SIZING TOOL

If you want to try out the MDI Sizing Tool, you can refer to the official documentation here: <https://learn.microsoft.com/en-us/defender-for-identity/deploy/capacity-planning>.

If you have more than 350 servers to install the MDI sensor on, you need to contact Microsoft Support.

Manual sizing estimation for domain controllers

If you can't use the MDI Sizing Tool, you can manually gather data to estimate your domain controller's capacity. Here's a step-by-step guide to help you gather the data:

Step 1: Plan data collection

- **Objective:** Collect the packets/sec performance counter from all your domain controllers
- **Duration:** Over a 24-hour period
- **Interval:** Use a 5-second sampling interval to capture detailed data

Step 2: Use Performance Monitor to collect data

1. Open **Performance Monitor** by pressing **Win + R**, type **perfmon**, and press **Enter**.
2. In the left pane, expand **Data Collector Sets**.
3. Right-click on **User Defined** and select **New > Data Collector Set**.
4. Enter a meaningful name (e.g., **MDI DC Packet Capture**) and select **Create manually (Advanced)**, then click **Next**.

5. Check **Create data logs** and select **Performance counter**, then click **Next**.
6. Click **Add** to add a performance counter.
7. In the **Available counters** list, scroll to **Network Adapter**.
8. Expand **Network Adapter** and select **Packets/sec**.
9. If unsure about the specific adapter, select all instances and then click **Add** and **OK**.
10. Change the sample interval to **5 seconds**, then click on **Next**.
11. Choose a folder where the data will be saved, then click **Next**.
12. Select **Start this data collector set now**.
13. Click **Finish**.

Step 3: Collect data over 24 hours

Allow the data collector set to run for a full 24 hours to capture both peak and off-peak network activity.

Step 4: Stop the data collector set

1. Return to **Performance Monitor**.
2. Expand **Data Collector Sets > User Defined**.
3. Right-click your data collector set (e.g., **MDI DC Packet Capture**) and select **Stop**.

Step 5: Analyze the collected data

1. Navigate to the folder where the data was saved.
2. Look for the file with a **.blk** extension, then double-click that file to open it in **Performance Monitor**.
3. In **Performance Monitor**, select the **Packets/sec** counter.
4. Analyze the graph to identify the following:
 - Average packets/sec over 24 hours
 - Maximum packets/sec during peak periods

Step 6: Calculate averages

1. Note the overall average value displayed in **Performance Monitor**.
2. Zoom into the busiest 15-minute window on the graph.
3. Calculate the average and maximum during this period.

Step 7: Evaluate domain controller capacity

In *Table 2.6*, you will see the CPU and RAM needed for the sensor server:

Busy packets/second	CPU (physical cores)	RAM (GB)

0-1k	0.25	2.50
1k-5k	0.75	6.00
5k-10k	1.00	6.50
10k-20k	2.00	9.00
20k-50k	3.50	9.50
50k-75k	5.50	11.50
75k-100k	7.50	13.50

Table 2.6 – Estimated CPU and RAM needed for the MDI sensor (DC only)

As you can see in *Figure 2.2*, the result from the 24-hour collection was as follows:

- **Average : 33**
- **Maximum : 1581**

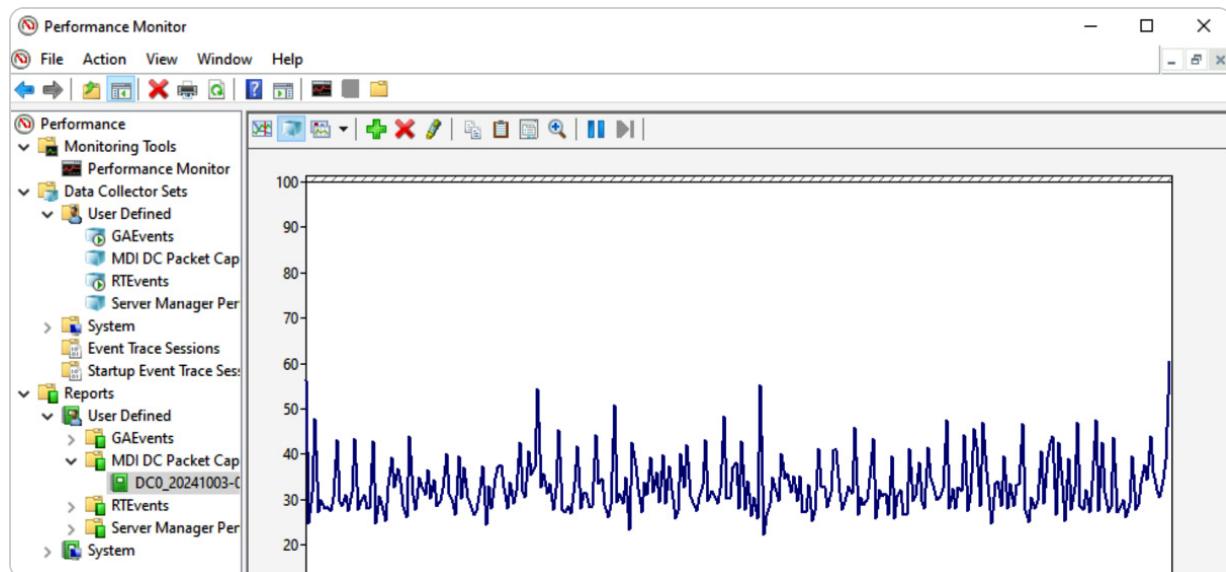


Figure 2.2 – Performance Monitor

This means that in our small lab we need at least 0.75 for the CPU and 6 GB of RAM. Be aware that this data collection is just for the MDI sensor and not for the overall capacity of the domain controller.

Let's see the prerequisites for AD FS and AD CS before we can install the MDI sensor on those servers.

Prerequisites for AD FS and AD CS

Primarily, when integrating AD FS and AD CS with an MDI sensor, the types of servers compatible with the MDI sensor include the following:

- Federation servers
- Certificate servers

In other words, it isn't required in **Web Application Proxy (WAP)** and is not supported for offline-certificate servers for obvious reasons.

WHY MDI ISN'T REQUIRED FOR WAP SERVERS

WAP servers don't need the MDI sensor because these servers primarily act as gateways, forwarding authentication requests to backend services such as AD FS. Since WAP servers don't directly process authentication or authorization requests, MDI focuses its monitoring on domain controllers and AD FS servers, where these critical operations take place.

The other prerequisites for the MDI sensor are the same as for our domain controllers.

For AD FS, we need to configure advanced auditing settings. The MDI sensor is set to automatically gather syslog events, and to detect potential security threats through MDI, it specifically uses and needs certain Windows event logs that the sensor extracts/parses from the domain controllers.

These are the events that are required from AD FS:

- **1202** : The federation service validated a new credential
- **1203** : The federation service failed to validate a new credential
- **4624** : An account was successfully logged on
- **4625** : An account failed to log on

These are the events that are required from AD CS:

- **4870** : Certificate Services revoked a certificate
- **4882** : The security permissions for Certificate Services changed
- **4885** : The audit filter for Certificate Services changed
- **4887** : Certificate Services approved a certificate request and issued a certificate
- **4888** : Certificate Services denied a certificate request
- **4890** : The certificate manager settings for Certificate Services changed
- **4896** : One or more rows have been deleted from the certificate database

We will configure these servers later in the book, but do familiarize yourself with the different types of Defender tables in the Defender XDR portal. You will be running these KQL queries later to see whether you are receiving any data.

The first query checks whether we have any rows that include the word `Adfs` in the `Protocol` column from the table called `IdentityLogonEvents` :

```
IdentityLogonEvents | where Protocol contains 'Adfs'
```

The second query checks whether we have any rows that equal (==) the word `Adcs` in the `Protocol` column from the table called `IdentityDirectoryEvents` :

```
IdentityDirectoryEvents | where Protocol == "Adcs"
```

Active Directory service accounts

Active Directory service accounts are specialized accounts created to operate applications, services, and scheduled tasks. These accounts are essential for automated operations that require specific privileges within Windows environments, and they need careful management to maintain security and efficiency.

The types of service accounts include the following:

- **Standalone managed service accounts (sMSA)** : This is a type of Active Directory account designed to provide automatic password management and simplified **service principal name (SPN)** management for services running on individual Windows servers. sMSAs enhance security by automating password changes without administrator intervention, thus reducing the risk of password leaks or unauthorized access. Each sMSA is tied to a single computer and cannot be used on multiple systems, making it ideal for services that need isolated, managed credentials on a single server.
- **Group managed service accounts (gMSAs)** : These provide automated password management and simplified SPN management for services running across multiple Windows servers within a domain. Designed to extend the capabilities of sMSAs, gMSAs allow services to operate under the same account name across different systems, facilitating easier management of services in a load-balanced environment. This feature significantly reduces the administrative burden associated with password management and enhances security by ensuring complex, regularly changed passwords without manual intervention.
- **Computer accounts** : Utilizing a computer account is a suitable alternative in the choice of managed service accounts. The LocalSystem account, a pre-configured local account, holds extensive permissions locally and represents the computer's identity on the network. Services operating under the LocalSystem account access network resources with the credentials formatted as `<domain_name>\<computer_name>`, using the computer account's identity. Known by its predefined name, `NT AUTHORITY\SYSTEM`, the LocalSystem account can be used to initiate a service and establish a security context for it.
- **User accounts** : A user account can be a local user account in a specific host or a domain user account. The local user account (formatted as `.\\UserName`) only exists within the host computer's Security Account Manager database and lacks an Active Directory object. It cannot be authenticated by the domain, restricting the service to anonymous-only network access and preventing support for Kerberos mutual authentication. Therefore, local user accounts are generally unsuitable for directory-enabled services. A domain user account leverages Windows and Microsoft Active Directory Domain Services security features, granting the service both local and network permissions, as well as permissions from any groups the account belongs to. This account type supports Kerberos mutual authentication.

Why is gMSA recommended?

A gMSA provides automated password management, which is a significant advantage over traditional service accounts, where passwords need manual updates and can be a security risk if not handled correctly. Here are some reasons why gMSA is recommended for MDI:

- **Security** : gMSAs improve security by automatically managing the account's password. This reduces the risk of password theft or misuse, as the passwords are complex and changed regularly without administrative intervention. The passwords are 120 characters long and automatically updated every 30 days, and Key Distribution Service (KDS) is responsible for handling the password.
- **Simplified management** : gMSAs eliminate the need for manual password management, reducing administrative overhead and minimizing human errors in password updates. The passwords are not affected by any password policies or fine-grained password policies.
- **Scalability and reliability** : gMSAs can be used across multiple servers for services that are load balanced. This is particularly useful in large deployments or environments where MDI sensors are installed on multiple domain controllers, ensuring consistent service account configurations across the board.

In a standard Active Directory environment that includes AD FS, AD CS, and domain controllers, it is advisable to use multiple gMSAs to enhance security. Specifically, using separate gMSAs for different roles – one for domain controllers, one for AD FS servers, and one for AD CS servers – provides an additional layer of security.

By isolating service accounts with separate gMSAs for each role, you limit the potential damage in the event of a compromise. If an attacker gains control of one gMSA, their access is confined to the specific servers and services associated with that account, reducing the risk of a widespread breach. This isolation ensures that even if one gMSA is compromised, the attacker cannot easily move laterally to other critical systems, thereby containing the breach.

Each gMSA is configured with only the permissions necessary for the specific service it supports, adhering to the principle of least privilege. This approach reduces the attack surface and ensures that service accounts have minimal access, further enhancing security.

In the case of a security incident, having separate gMSAs allows for a more targeted and efficient response. You can quickly rotate credentials or disable the compromised gMSA without disrupting other critical services, maintaining operational stability while addressing the security issue.

However, while gMSAs significantly enhance security, they are not entirely immune to attacks. Attackers can potentially take over a gMSA if they gain administrative access to a domain controller, which allows them to retrieve the gMSA's password. Another attack vector involves compromising the systems or users that have permission to use the gMSA. If an attacker compromises a system or user with access to the gMSA, they could potentially execute actions as that gMSA, thereby gaining access to sensitive systems or data. Therefore, it's crucial to ensure that the security of the systems managing gMSAs is maintained at the highest level, with regular audits, monitoring, and strict access controls in place.

By leveraging multiple gMSAs for the Active Directory service account used by MDI, organizations can significantly enhance their security posture, minimize the risk of lateral movement in the event of a compromise, and ensure that their security practices align with industry best practices for both security and operational efficiency.

Active Directory service accounts in MDI

Active Directory Service Accounts (DSA) are a specialized account used by the MDI sensor to interact with your **Active Directory (AD)** environment. This account performs various operations necessary for MDI to function effectively, such as querying AD for user and group details, reading logon events, and other activities that help MDI detect and analyze potential threats within the network.

It is highly recommended to configure a DSA to get the full security coverage, but it is also optional. If you are already planning to install the MDI sensor on AD FS or AD CS, then a DSA is required. The DSA will also be able to take care of the following:

- Domain and trust mapping is performed at sensor startup and subsequently every 10 minutes to ensure current configurations are recognized.
- Querying details via LDAP from another domain is done when activities involving entities from those domains are detected, allowing for a thorough understanding of cross-domain interactions.
- Requesting member lists of local administrator groups through a **SAM-R call** to devices observed in network traffic, events, and Event Tracing for Windows (ETW) activities. This data is crucial for identifying potential paths for lateral movement.
- Accessing the **DeletedObjects** container to gather data on deleted users and computers, providing insights into historical changes.

Regarding the **DeletedObjects** container, if you haven't already enabled the Recycle Bin feature in Active Directory, it's recommended that you do so to facilitate the recovery of deleted objects. To enable this feature, log in to a domain controller within your Active Directory forest and open Windows PowerShell. Then, execute the following command:

```
Enable-ADOptionalFeature -Identity 'Recycle Bin Feature' -Scope ForestOrConfigurationSet -Target (Get-ADDomain).DNSRoot -Confirm:$false
```

We can configure a DSA in two ways: as a regular user account within Active Directory or using a gMSA, and it is the latter option that's highly recommended to configure. A gMSA is a type of account that automatically manages its password and simplifies the management of **service principal names (SPNs)**, providing a more secure and efficient way to manage service accounts.

Now that we've covered all the prerequisites and gained a solid understanding of how the MDI sensor operates, it's time to move on to the deployment process.

Deployment of MDI – a step-by-step guide

Throughout the pre-installation phase and our thorough planning checklist, we've dissected the inner workings of the MDI sensor and the services it integrates with. We've explored the necessary licensing requirements and delved into other critical needs such as networking configurations and the essential DSAetup. So, before we dive into the next step, I hope that you have prepared your environment.

We will now go through the installation process, including how we can configure the sensor and verifying that we have successful onboarding and full functionality:

1. **Installing the MDI sensor** : With the preparations complete, you can now begin the installation of the MDI sensor on your designated servers. Follow the instructions provided in the next section of this chapter to download and install the sensor.
2. **Configuring the MDI sensor** : After installation, we will look at the different configurations that can be done for the MDI sensor to tailor its operations to your specific environment.
3. **Validation** : Once the MDI sensor is installed and configured, proceed with validation tests to ensure it is functioning as expected. Monitor the sensor's output and verify that it is correctly reporting and responding to the simulated threat scenarios.

DID YOU REMEMBER?

Microsoft .NET Framework 4.7 or later needs to be installed on the sensor server; if you forgot it, the Defender for Identity sensor setup package will install it, which potentially requires a server reboot. Make sure that you have a maintenance window scheduled so that you can do a reboot after the installation.

As we move forward together in setting up MDI, it's time to get hands-on with your own lab environment. In the next sections, we'll guide you through obtaining the MDI installation package and securing your access key – both essential steps in our journey. By working through these steps in your lab, you'll gain practical experience and be well prepared for the full deployment. Let's dive in and get started!

Following with your own lab environment

If you hold an Azure subscription or have a lab setup at home, you are well equipped to follow along with the setup and configuration of this infrastructure environment. For those utilizing Microsoft Azure, I have constructed an Azure Bicep template to streamline the deployment process. This template enables you to deploy and automatically configure key components with ease.

This template includes the setup of the following:

- Six Azure virtual machines:
 - One Active Directory Domain Services server
 - One Active Directory Federation Services server
 - One Active Directory Certificate Services server
 - One Entra Connect server

- One Linux syslog servers
- One Linux TinyProxy servers
- One virtual network with five subnets:
 - Active Directory Subnet
 - Bastion Subnet
 - DMZ Subnet (used for TinyProxy)
 - Server Subnet
 - Client Subnet (reservation for future use)
- One NAT gateway:
 - For outbound internet connectivity
- One Log Analytics workspace:
 - To receive syslog messages

The autoconfiguration will help you with the following:

- Setting up a new Active Directory forest and domain
- Setting up an AD FS farm
- Setting up AD CS
- Setting up Entra Connect, but not connecting the server to Entra ID
- Azure Monitor data collection rules for syslog and custom **json** files
- Configuring syslog servers with Azure Monitor Agent
- Configuring TinyProxy servers

By utilizing this template and configuration, you can effortlessly establish a robust and secure IT infrastructure tailored for optimal performance and security. Deployment through the ARM template not only saves time but also aligns with some best practices, ensuring a solid foundation from the start. Whether you're setting up for a test environment or a full-scale production deployment, this guide and the accompanying Azure template provide the necessary tools and instructions to get your infrastructure up and running efficiently.

So, your call to action is to deploy the template located in the GitHub repository for this chapter.

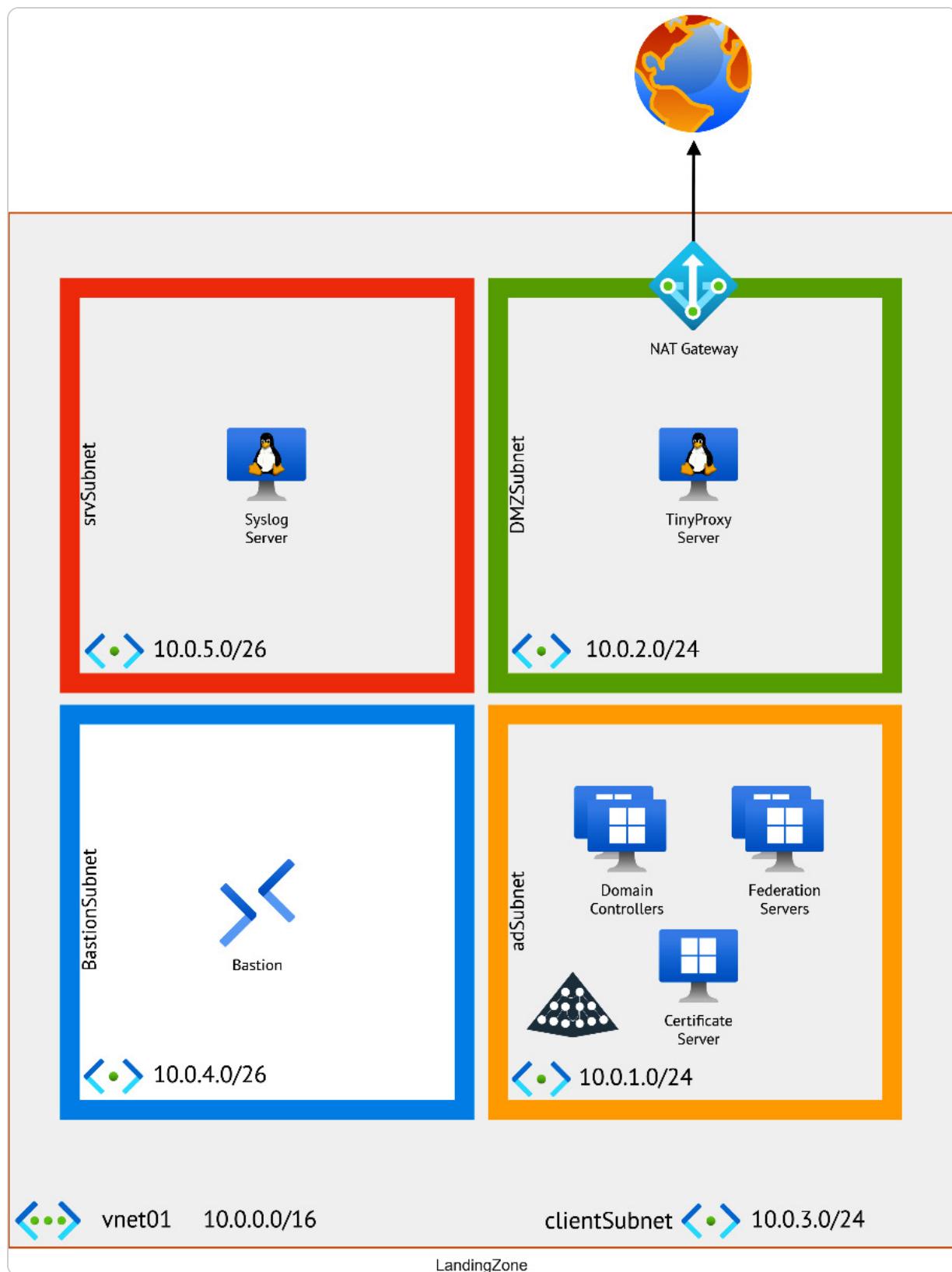


Figure 2.3 – Lab architecture

Getting the MDI installation package and access key

Before you can go ahead and install the MDI sensor, you must first download the latest version of the installation package, which can be found in the Microsoft Defender XDR portal:

1. **Access the portal** : Log in to security.microsoft.com using a highly privileged account, such as Global Administrator or Security Administrator.
2. **Navigate to the sensor installation page** : You have two options to reach the **Add a new sensor** page:
 - Directly via [https://security.microsoft.com/settings/identities? tabid=sensor](https://security.microsoft.com/settings/identities?tabid=sensor)
 - Alternatively, navigate through the portal by selecting **System > Settings > Identities > General > Sensors** and then click on **Add sensor**
3. **Download the package and get the access key** : From the **Add a new sensor** page, you can download the installer and obtain the access key necessary for the sensor installation (see *Figure 2.4*).
4. Transfer the package using secure methods, such as a temporarily secure and access-controlled file share.

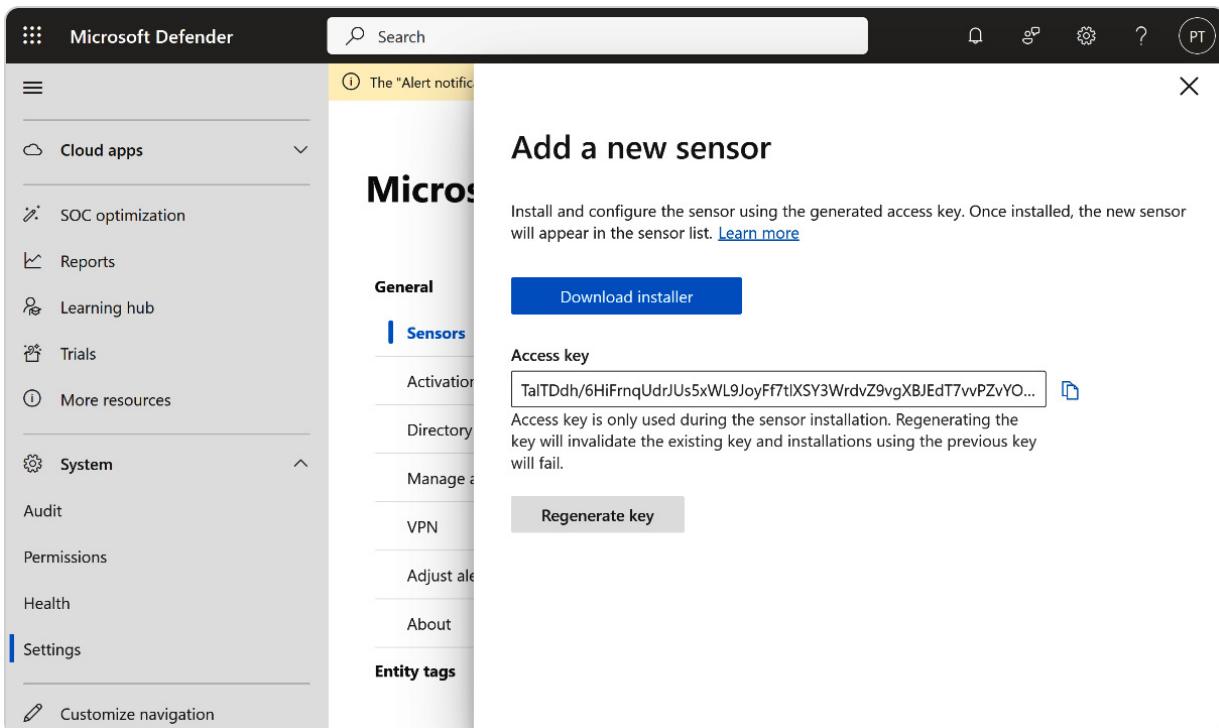


Figure 2.4 – MDI sensor access key

POPUP BLOCKED

Make sure to allow popups for the URL security.microsoft.com so you can continue the download of the package.

The download package is a `.zip` file containing the following:

- Npcap OEM version 1.0
- The Defender for Identity sensor installer (**Azure ATP Sensor Setup.exe**)
- A **.json** configuration setting file with connection information to your MDI instance

NCAP – TRANSITION FROM WINPCAP TO NPCAP IN MDI

PCAP, standing for **packet capture**, is essential for analyzing network traffic, which is crucial for security applications such as MDI. Historically, MDI utilized **WinPcap**, the standard packet capture library for Windows. However, as of version 2.184 released in mid-2022, the MDI installation package transitioned to using **Npcap 1.0 OEM** instead of WinPcap.

This change to Npcap was implemented to take advantage of its updated capabilities and enhanced performance features. Npcap offers better support for the latest Windows versions, includes capabilities for capturing loopback traffic and sending raw packets, and operates in a more secure driver mode. These enhancements make Npcap more suitable for modern security environments, providing robust support for MDI sensors to monitor and analyze network traffic, thereby improving threat detection and response capabilities efficiently and securely.

If you haven't been updating the MDI sensor since mid-2022, you will most definitely have an open health issue – **Sensor has issues with packet capturing component**. Uninstall the MDI sensor and WinPcap (if it was manually installed prior to the sensor installation) and then install the newest version of the MDI sensor.

The **.json** file looks as follows:

```
{
  "$type": "SensorInstallationConfiguration",
  "WorkspaceApplicationSensorApiEndpoint": {
    "$type": "EndpointData",
    "Address": "<your-workspace-name>sensorapi.atp.azure.com",
    "Port": 443
  },
  "WorkspaceId": "<<GUID>>"
}
```

The installer will check whether the target machine is a domain controller, a federation server (AD FS), certificate server (AD CS), Entra Connect server, or a dedicated standalone server (which will get the standalone sensor installation).

NOTE

The silent installation is configured to automatically restart the server; make sure that you have a maintenance window scheduled so that you can do a reboot after the installation.

Installing the sensor with a UI

With the **.zip** package transferred to the machines we are now ready for installation.

These are the prerequisites:

- Connectivity to the MDI cloud service
- Minimum MDI sensor requirements

- Local administrator privileges

The first thing we need to do is to extract the installation files from the . zip package:

1. Execute the **Azure ATP sensor setup.exe** file with elevated privileges by selecting **Run as administrator**.
2. On the **Welcome** page, choose your preferred language and click **Next**.
3. For **Sensor deployment type**, verify that the sensor type matches the role of the server:
 - **Sensor** : AD CS/AD DS/AD FS
 - Standalone sensor
4. On the **Configure the sensor** screen, specify the installation path and enter the access key that we collected earlier:
 - **Default installation path** : %programfiles%\Azure Advanced Threat Protection sensor
5. Click **Install** and wait for the installer to finish.
6. Verify the installed sensor version by looking at the file version attribute of the installer package and in the Defender XDR portal under the server sensor details after successful installation.

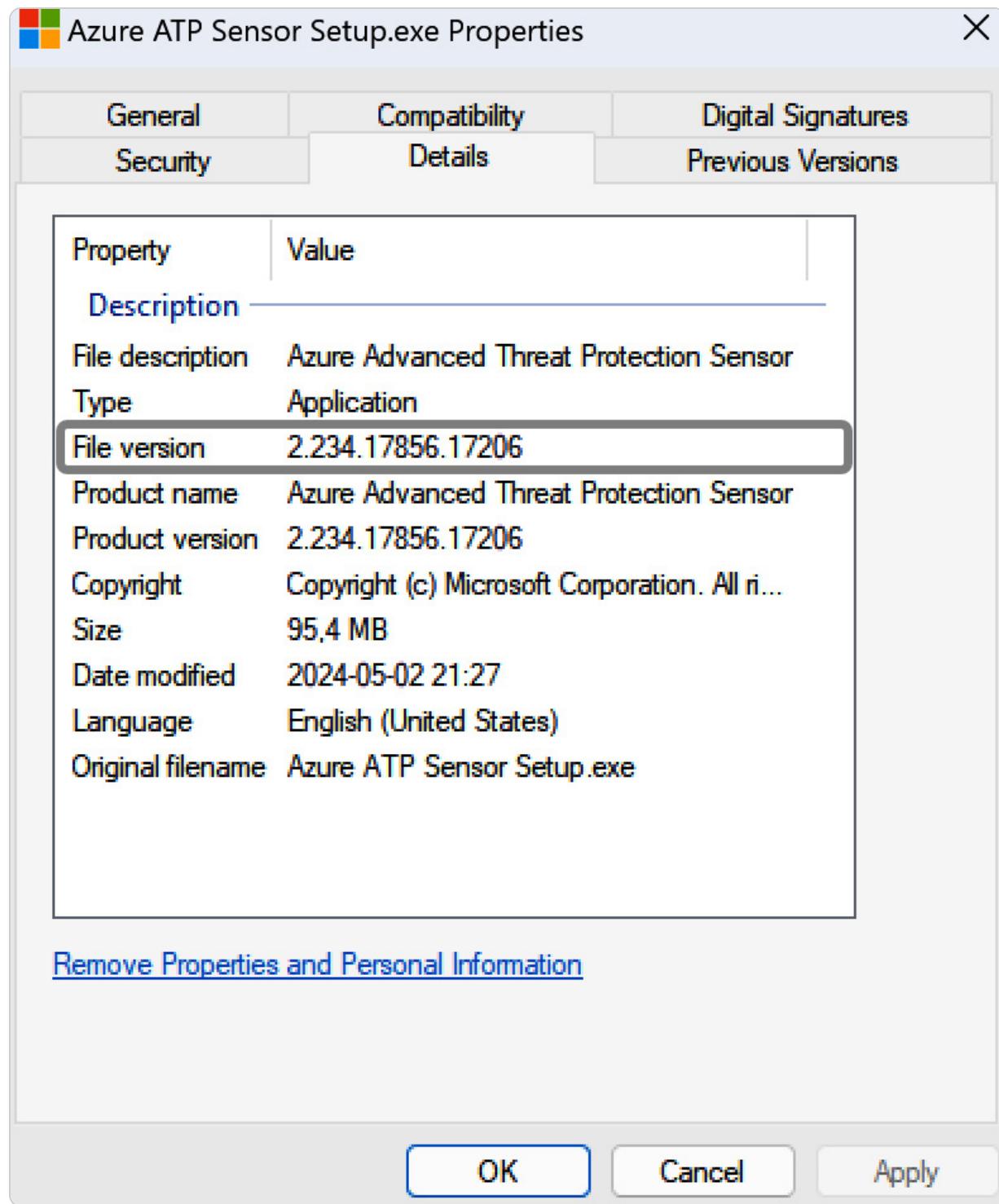


Figure 2.5 – Details for the Azure ATP sensor setup.exe file

You will see the same version of the MDI sensor in the Defender XDR portal on each of the devices; see *Figure 2.6*.

↑ ↓ ×

 **DC01**
● Medium

Sensor details ^

Service status	Created
Unknown	Feb 14, 2024 4:14 PM
Domain	Version
contoso.local	2.234.17856.17206
Update status	Delayed update
Unreachable	Disabled

Open health issues (1) ^

Issue	Severity	Generation time
Sensor stopped communicating	■■■ Medium	Apr 28, 2024 6:07 PM

Closed health issues (4) ^

Issue	Severity	Generation time

Manage sensor

Figure 2.6 – Sensor details in the Defender XDR portal

Installing the sensor with PowerShell

With the `.zip` package transferred to the machines we are now ready for installation.

These are the prerequisites:

- Connectivity to the MDI cloud service
- Minimum MDI sensor requirements
- Local administrator privileges
- Windows PowerShell 5.0 or later:
 - This is because the `Expand-Archive` cmdlet was introduced

We have several options when we install the MDI sensor through PowerShell. Make sure to understand the options and the syntax used together with the `.exe` file:

Options	PowerShell syntax
The default installation path	<code>%programfiles%\Azure Advanced Threat Protection Sensor</code>
Quiet installation	<code>/quiet</code>
Silent installation of .NET Framework	<code>NetFrameworkCommandLineArguments="/q"</code>
Access key	<code>AccessKey="**"</code>
Access key file	<code>AccessKeyFile=""</code>
Delayed update	<code>DelayedUpdate=true</code>
<code>LogsPath</code>	<code>%programfiles%\Azure Advanced Threat Protection Sensor</code>
<code>ProxyUrl</code>	<code>ProxyUrl="http://proxy.contoso.local:8080"</code>
<code>ProxyUserName</code>	<code>ProxyUserName="DOMAIN\User"</code>
<code>ProxyUserPassword</code>	<code>ProxyUserPassword="P@ssw0rd"</code>

Table 2.7 – PowerShell options and syntax for customizing the MDI sensor installation

Follow these steps for installing an MDI sensor without the UI:

1. Extract the installation files from the `.zip` package with the `Expand-Archive` cmdlet (adjust the different paths according to your needs):

```
$PathToInstallZipPackage = "C:\temp\Azure ATP Sensor Setup.zip"
$PathForDestinationFiles = "C:\install\MDI_sensor"
Expand-Archive -Path $PathToInstallZipPackage -DestinationPath
$PathForDestinationFiles
```

2. Save the access key that we collected earlier in a variable called `$AccessKey`, and then run the installer with the arguments that you have decided on (see *Table 2.7*):

```
$AccessKey = "<<Fill in the access key>>"
Set-Location $PathForDestinationFiles
.\\"Azure ATP sensor Setup.exe" /quiet NetFrameworkCommandLineArguments="/q"
AccessKey="$AccessKey"
```

3. Look at the installation logs to check for success or failure:

- **Location folder:** `%LocalAppData%\Temp`
- **Filename:** `Azure Advanced Threat Protection Sensor_YYYYMMDDHHMMSS_000_MsiPackage`

4. Verify the installed sensor version by looking at the file version attribute of the installer package and in the Defender XDR portal under the server sensor details after successful installation.

5. Use the following PowerShell script to check for successful installation and the output of the sensor version:

```
$MDIInstallLog = Get-ChildItem -Path "$env:LocalAppData\Temp" -Filter "Azure
Advanced Threat Protection Sensor_*_MsiPackage.log" -Recurse -ErrorAction
SilentlyContinue | Sort-Object -Property LastWriteTime -Descending | Select-
Object -First 1
if ($MDIInstallLog -eq $null) {
    Write-Host "MDI installation log file not found"
}
$MDIInstallCode = Get-Content -Path $MDIInstallLog.FullName | Select-String -
Pattern "Installation completed successfully"
if ($MDIInstallCode -eq $null) {
    Write-Host "MDI installation code not found"
}
$MDIProductVersion = Get-Content -Path $MDIInstallLog.FullName | Select-String -
Pattern "Product Version:"
if ($MDIProductVersion -match '\d+\.\d+\.\d+\.\d+') {
    $productVersion = $matches[0]
    Write-Host "Product version found."
} else {
    Write-Host "Product version not found."
}
Write-Host "MDI installation code: $MDIInstallCode"
Write-Host "MDI product version: $productVersion"
```

Let's look at a few examples of PowerShell syntax:

- PowerShell syntax for silent MDI sensor installation together with silent .NET Framework installation:

```
.\\"Azure ATP sensor Setup.exe" /quiet NetFrameworkCommandLineArguments="/q"
```

```
AccessKey="<Access Key>"
```

- PowerShell line with silent installation of .NET Framework and **AccessKeyFile** :

```
.\"Azure ATP sensor Setup.exe" /quiet NetFrameworkCommandLineArguments="/q"
AccessKeyFile="C:\temp\AccessKeyFile.txt"
```

- PowerShell line with proxy configuration:

```
.\"Azure ATP sensor Setup.exe" /quiet ProxyUrl="http://proxy.contoso.local"
[ProxyUserName="domain\proxyuser"] [ProxyUserPassword="Password"]
```

Installation complete! Great job. Now, let's shift our focus to configuring MDI to communicate securely with Microsoft cloud services through a proxy solution.

Navigating step-by-step proxy configuration for MDI

The significance of integrating firewalls into your IT infrastructure cannot be overstated. In recent years, particularly since 2020, there has been a rapid increase in the vulnerabilities and risks associated with IT products and services. As a strategic response, many organizations are opting to regain a measure of security control by adopting proxy solutions.

Having a proxy in an IT environment is crucial for enhancing security and managing network traffic. It acts as an intermediary, ensuring that external systems communicate through it, rather than directly with your network's servers. This setup allows for better control over data flow, provides an additional layer of security by filtering harmful traffic, and enables more efficient network performance through caching. It's an essential component for maintaining a secure and efficient network infrastructure.

In the digital age, where cyber threats loom large, the security of communication channels is paramount, especially when it involves sensitive data exchange with cloud services. MDI is designed to help protect your enterprise's identities from compromise, but its efficacy can be significantly enhanced with the right proxy configuration. This is where **TinyProxy**, as an example, steps in – a lightweight yet powerful proxy solution that ensures secure and continuous communication with Microsoft cloud services, not just for MDI but also Microsoft/Windows Update and Defender for Endpoint, for example. There are also other proxy solutions available, such as Squid or any forward proxy server, that can fulfill the same role.

TinyProxy serves as a gatekeeper, managing and directing the flow of traffic between MDI and the cloud. Its architecture is designed for efficiency, requiring minimal resources to run, which makes it an ideal choice for organizations looking to implement a proxy without significant overhead. By encrypting data flow and providing controlled access through diligent filter settings, TinyProxy ensures that only authorized URLs are permitted, thereby strengthening the security posture – which will soon be the most important task you share with other IT professionals in your company.

The strategic benefits of integrating TinyProxy with MDI are several:

- Firstly, it provides an additional layer of defense, encrypting communication and safeguarding against potential eavesdropping.
- Secondly, its filtering capabilities mean that only necessary traffic reaches MDI, reducing noise and allowing for more accurate threat detection.
- Moreover, TinyProxy's compatibility with MDI allows for a seamless setup, ensuring that the proxy acts in concert with MDI's threat detection mechanisms, rather than as a hindrance.

Setting up a proxy for MDI or any of the other Microsoft cloud services, such as all the Azure services or the other Defender services, does come with its challenges, such as ensuring uninterrupted service and maintaining up-to-date filter lists. If you have ever looked at the IP list that Microsoft provides of the different types of services, it can feel a bit overwhelming. Best practices suggest a thorough planning phase, where the necessary URLs and domains for MDI communication are whitelisted. Regular maintenance and monitoring are also crucial to adapt to the evolving threat landscape and to update the filters accordingly.

In practice, a well-configured TinyProxy can significantly enhance MDI's ability to detect and respond to threats. For instance, in a scenario where an attacker attempts to exfiltrate data, TinyProxy's encrypted channels would prevent unauthorized access, while its filtering system would block communication to malicious domains. We will learn how to send this type of information to a Log Analytics workspace where we can be alerted if we have systems that want to talk to malicious domains, and we can also get more context with Microsoft Sentinel and connected Threat Intelligence.

As we learned in the *Networking* section, we need to allow a few URLs for our servers to be able to send their data and telemetry to the MDI service.

Installing TinyProxy

To install TinyProxy on an Ubuntu 22.04 minimal server installation, you will need to follow these steps:

1. Log in to or access your TinyProxy server.
2. Update your package list with the following command:

```
sudo apt update
```

3. Once the package list is updated, upgrade your system with this:

```
sudo apt upgrade
```

This step is optional but recommended to ensure you have the latest versions of installed packages.

4. Install TinyProxy using the following command:

```
sudo apt install tinyproxy
```

5. After installation, you'll need to configure TinyProxy. The configuration file is located here:

```
/etc/tinyproxy/tinyproxy.conf
```

You can edit this file with a text editor of your choice, for example, using nano:

```
sudo nano /etc/tinyproxy/tinyproxy.conf
```

6. In the configuration file, you can set various parameters, such as the port TinyProxy listens on, which by default is port **8888** .

You can change this if necessary. There's more about the configuration file in the *Configuring TinyProxy* section.

7. Set up Allow/Deny rules in the configuration file to control which clients can access your proxy server.

8. Once you have configured TinyProxy to your liking, save the changes (*Ctrl + O*) and exit the text editor (*Ctrl + X*) .

9. Restart TinyProxy to apply the changes with the following command:

```
sudo systemctl restart tinyproxy
```

10. You can also enable TinyProxy to start on boot with the following:

```
sudo systemctl enable tinyproxy
```

Now that we have installed TinyProxy on our Ubuntu server, it's time to start configuring it so our MDI-supported systems can talk to the MDI cloud service without any disruptions.

Configuring TinyProxy

To configure TinyProxy, we need a few things:

- The IP addresses or subnets that will be allowed to talk to the proxy servers
- All the URLs for the MDI service
- The log level and whether you want the logs to go to file or syslog – these are the log levels:
 - Critical (least verbose)
 - Error
 - Warning
 - Notice
 - Connect (log connections without Info's noise)
 - Info (most verbose)

Now that we have installed TinyProxy and have an up-to-date Ubuntu system, it's time to configure TinyProxy.

You can look at the GitHub repository of this chapter for a reference `tinyproxy.conf` file.

Open the `/etc/tinyproxy/tinyproxy.conf` file with the text editor of your choice – here we use nano:

```
sudo nano /etc/tinyproxy/tinyproxy.conf
```

Now you can start forming the configuration file for your needs.

Ensuring success with post-installation activities

In this section, we will configure the activities required for a successful implementation of MDI. After the initial installation of the MDI sensor, certain configurations and best practices must be established to optimize the functionality and security of your deployment. This involves setting up DSAs with gMSAs, configuring **Security Account Manager-Remote (SAM-R)** protocol settings, and implementing RBAC to ensure proper access management.

The post-installation process involves several key steps that go beyond basic setup, focusing on robust account management, secure communication protocols, and detailed access control measures. By carefully configuring these elements, you can create a resilient environment that supports proactive threat detection and enhances your overall security posture. Here's what we'll cover in this section:

1. **AD DSAs** : To begin, we'll guide you through the process of creating and configuring DSAs and gMSAs. These accounts are pivotal for the operation of MDI sensors, providing the necessary permissions to perform tasks such as scanning for security threats and anomalies within your network. We will provide detailed steps on how to correctly set up these accounts, including the automation of password management for gMSAs, which is crucial for maintaining security integrity and operational efficiency.
2. **SAM-R** : Next, we will address the configuration of the SAM-R protocol. This configuration is essential for allowing MDI to query account information securely from your domain controllers, which is vital for accurate monitoring and threat detection.
3. **RBAC** : Lastly, we will cover best practices for implementing RBAC within your MDI environment in the Defender XDR portal. RBAC is a critical component for managing access rights and ensuring that only authorized personnel have control over specific functions within MDI. Proper implementation of RBAC not only secures your environment but also helps in complying with various regulatory requirements by ensuring that access is granted based on the principle of least privilege.

DSAs

Do you remember from earlier that our account only requires read permission? So do all the objects within Active Directory, which includes the deleted-objects container (so that MDI can be able to detect when users are deleted).

THE RIGHT TYPE OF AD SA

We will only focus on gMSAs because running something else is not a security best practice. Apart from gMSA support, you could use a regular user account with a password, but as you might expect, this is a less secure alternative and can lead to downtime if the password expires.

Before we create our gMSA account, we will learn about and check a key component that handles gMSA.

The **Key Distribution Service (KDS)** root key is an essential component required for the proper functioning of gMSAs in Active Directory. This root key is used by KDS to generate passwords for gMSAs, enabling the automatic management of these passwords.

Why we need the KDS root key for gMSAs

Setting up a KDS root key is a preliminary step before deploying gMSAs. Once established, it allows your organization to utilize gMSAs efficiently, bolstering security protocols and reducing the administrative burden associated with manual password updates:

- **Automatic password management** : The KDS root key allows for the secure generation and distribution of complex passwords for gMSAs. These passwords are automatically updated by Active Directory, ensuring that the accounts remain secure without requiring manual intervention.
- **Security** : Automatically managed passwords reduce the risk of password leaks or unauthorized access, as the passwords are highly complex and are changed frequently according to policies defined by the administration.
- **Consistency across servers** : Since gMSAs can be used across multiple servers within a domain, the KDS root key ensures that all instances of a gMSA use the same password at any given time, simplifying credential management across different servers.

Now, let's check whether we need to create the KDS root key and then we can create our gMSA account.

KDS root key

To check whether we have an existing KDS root key, we need to run a PowerShell command when we are connected to a domain controller:

1. Connect to one of your domain controllers in a secure way (we need to think of our tiering).
2. Start Windows PowerShell from the start menu.
3. Type the following command and see whether you get any output:

```
Get-KdsRootKey
```

If the KDS root key does not exist, keep following the guide and commands.

4. In the same Windows PowerShell window, now type the following command, which will create a new KDS root key on the domain controller running the command, the KDS service can use the root key directly, but other domain controllers need to wait for the Active Directory replication:

```
Add-KdsRootKey -EffectiveImmediately
```

The crucial KDS root key has been successfully created and initiated for inclusion in the replication process. We will now shift our focus to creating our gMSA account.

gMSA

Now that we have verified and, for some of you, created the KDS root key, it's time to create our gMSA account:

1. Connect to one of your domain controllers in a secure way (we need to think of our tiering).
2. Start Windows PowerShell from the start menu.

3. Create new variables with the name of the gMSA account, the name of the sensor group that will contain all the computer accounts where we installed the MDI sensor, and then a list of all computer accounts (change this as needed):

```
$gMSA_Name = 'MDIGMSA'
$gMSA_HostsGroupName = 'MDIGroup'
$gMSA_HostNames = 'DC1', 'DC2', 'ADCS1', 'EntraConnect', 'ADFS1'
```

4. Now create the sensor group and add the computer accounts to the group:

```
$gMSA_HostsGroup = New-ADGroup -Name $gMSA_HostsGroupName -GroupScope DomainLocal
-PassThru
$gMSA_HostNames | ForEach-Object { Get-ADComputer -Identity $_ } | ForEach-Object
{ Add-ADGroupMember -Identity $gMSA_HostsGroupName -Members $_ }
```

5. Type the following command to create the gMSA account and add the sensor group so the computer accounts can retrieve the gMSA password:

```
New-ADServiceAccount -Name $gMSA_Name -DNSHostName ($gMSA_Name + "." +
$env:USERDNSDOMAIN) -Description "Microsoft Defender for Identity gMSA account" -
KerberosEncryptionType AES256 -PrincipalsAllowedToRetrieveManagedPassword
$gMSA_HostsGroupName
```

Using **AES256** with gMSA accounts is important for securing Active Directory environments. It provides stronger encryption, making it much harder for attackers to crack Kerberos tickets compared to older methods such as **RC4** or **DES**, which have known vulnerabilities. By setting **KerberosEncryptionType** to **AES256**, you ensure that your gMSA uses a higher standard of encryption, aligning with best practices and offering better protection against identity-based attacks.

6. To verify that the gMSA is created, run the following command:

```
Get-ADServiceAccount -Name $gMSA_Name
```

7. Verify that the domain controller can retrieve the password for the gMSA; the output we are looking for is **True** :

```
Test-ADServiceAccount
```

8. If you haven't already, we need to activate the **Recycle Bin** feature in Active Directory, which we can do with the following command:

```
Enable-ADOptionalFeature -Identity 'Recycle Bin Feature' -Scope
ForestOrConfigurationSet -Target (Get-ADDomain).DNSRoot -Confirm:$false
```

9. The gMSA account needs read permissions, so first we will capture the distinguished name of the deleted-objects container; we will then take ownership of that container before we finally grant the identity **List Contents** and **Read Property** permissions:

```
$distinguishedName = ([adsi]'').distinguishedName.Value
$deletedObjectsDN = 'CN=Deleted Objects,{0}' -f $distinguishedName
$params = @("$deletedObjectsDN", '/takeOwnership')
C:\Windows\System32\dsaccls.exe $params
$params = @("$deletedObjectsDN", '/G", ('{0}\{1}:LCRP' -f ([adsi]'').name.Value,
$Identity))
C:\Windows\System32\dsaccls.exe $params
```

We have now created our gMSA account and given it read permissions, so let's start looking at the SAM-R configuration.

Configuring SAM-R

To configure SAM-R, you need to have completed and configured the DSA for MDI because the ADSA (gMSA) needs to have permissions to be able to perform SAM-R queries for detecting lateral movement.

IMPORTANT NOTE

*Ensure that your **Group Policy object (GPO)** is applied to all computers, **excluding** domain controllers. It's highly recommended to perform the upcoming changes in **Audit Mode** before enforcing the policy in your production environment.*

*There have been posts and reports that the policy can break the creation of **offline address book (OAB)** in Microsoft Exchange 2013 and Microsoft Exchange 2016. While this example specifically mentions Microsoft Exchange Server, the issue similarly affects other applications that utilize **AuthZ** in the same way. Add the Exchange Servers universal group to the policy (see the following steps) to fix the issue.*

Follow these steps to create a new GPO for SAM-R:

1. Log in to one of the domain controllers (or the domain controller that holds the PDC Emulator FSMO role).
2. Open **Group Policy Editor** and create a new GPO or edit an existing GPO.
3. Navigate to **Computer Configuration > Policies > Windows settings > Security settings > Local Policies > Security options**.
4. Locate the policy named **Network access: Restrict clients allowed to make remote calls to SAM**.
5. Add the account that you previously created and the other accounts you found during the audit period.

You have now configured lateral movement path detection for devices connected to your Active Directory but in the modern world many of our devices will live in the cloud and in Entra ID as **Microsoft Entra-joined devices**.

To be able to detect lateral movement for our Entra-joined devices, we need to create a new Device profile within Microsoft Intune. We will first locate the MDI Directory Service account SID (security identifier) because we need that ID when we configure the new device profile:

1. Log in to one of your domain controllers in a secure way.
2. Start Windows PowerShell from the start menu.
3. Type the following command and grab the value of the SID:

```
Get-ADServiceAccount -Identity MDIGMSA | Select-Object Name, SID
```

4. Log in with an account that has the necessary permissions in Microsoft Intune admin center (<https://intune.microsoft.com>) to create a new device profile.
5. Click on **Devices**.

6. Locate the **Manage devices** menu and then click on **Configuration**.

The screenshot shows the Microsoft Intune admin center interface. The left sidebar contains navigation links such as Home, Dashboard, All services, Devices, Apps, Endpoint security, Reports, Users, Groups, Tenant administration, and Troubleshooting + support. The main content area is titled "Devices | Configuration". It features a search bar at the top, followed by a navigation bar with tabs: Policies (selected), Import ADMX, and Monitor. Below this is a toolbar with buttons for Create, Refresh, Export, and Columns. A message indicates "0 policies". The main pane displays a table with columns: Policy name, Platform, and Policy type. A large, semi-transparent hexagonal watermark is overlaid on the right side of the table.

Figure 2.7 – Intune device policy

7. In the **Policies** tab, click on **+ Create**, and then click on **New Policy**.

8. In the **Create a profile** wizard, choose the following values:

- **Platform** : Windows 10 or later
- **Profile type** : Settings catalog
- Click **Create** .

9. Enter an expressive name and description for your policy.

10. On the **Configuration settings** tab, click on **+ Add settings**.

11. In **Settings picker**, type **Network Access Restrict Clients Allowed To Make Remote Calls To SAM**, then click on **Search**.

12. Under **Browse by category**, click on **Local Policies Security Options**, and then select the **Network Access Restrict Clients Allowed To Make Remote Calls To SAM** setting under **Setting name**.

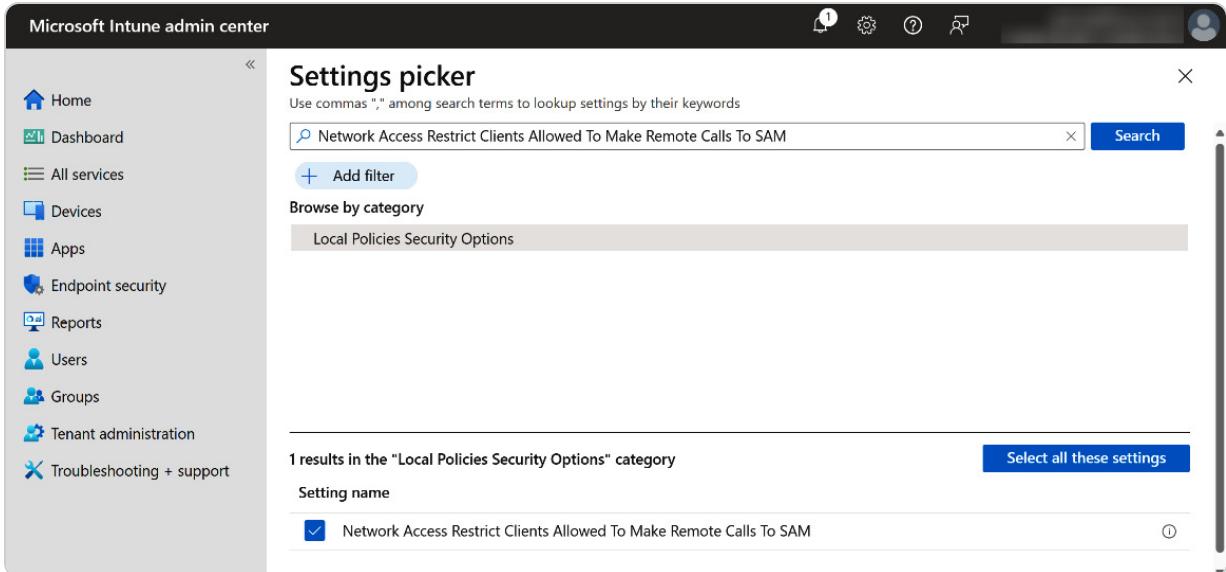


Figure 2.8 – Settings picker

13. Click on the **X** at the top right of the **Settings picker** menu to close the blade.

14. Enter the security descriptor (SDDL) in the field next to the setting name:

```
O:BAG:BAD:(A;;RC;;;BA)(A;;RC;;;%SID%)
O:BAG:BAD:(A;;RC;;;BA)(A;;RC;;;S-1-5-32-544)
```

15. Click **Next**.

16. Choose the **Scope tag** value of your choice, then click **Next**.

17. Choose the **Assignment** value that fits your environment, then click **Next**.

18. Review the new profile and when ready, click on **Create**.

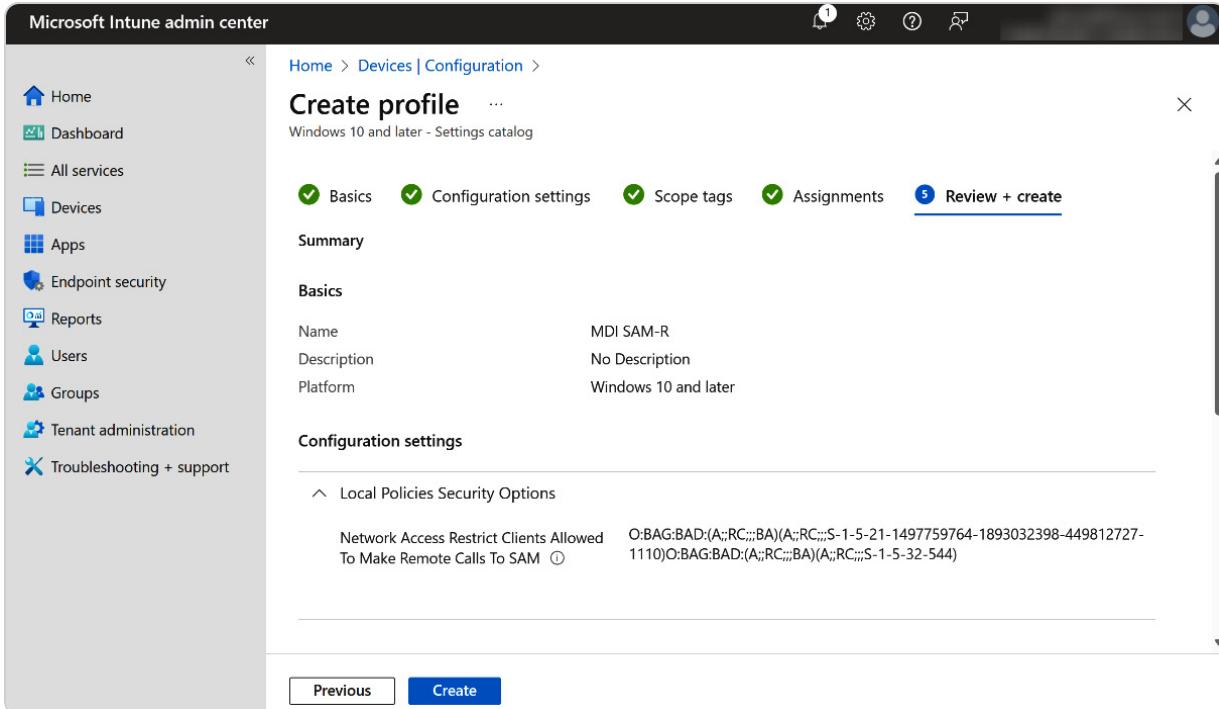


Figure 2.9 – Intune SAM-R device policy

We have now configured SAM-R policies both for our on-premises devices as well as for our cloud-only devices, and with that we are now ready to add our gMSA in the Defender XDR portal.

Setting the gMSA in the Defender XDR portal

The MDI sensors get the settings from the Defender XDR portal, so we need to configure our sensors as a last step.

With an account that has permissions in the Defender XDP portal, do the following:

1. Go to <https://security.microsoft.com>.
2. Sign in with your Security Administrator or Global Administrator account (keep in mind that you should be careful with the Global Administrator account).
3. Go to **System > Settings > Identities > Directory service accounts**.
4. Click on **+ Add credentials**.
5. Configure the following:
 - **Account name** : Your gMSA, such as **MDIGMSA\$** ; it is important that when we use a gMSA, we end the username with the \$ sign
 - **Group managed service account** : Toggle this on.
 - **Domain** : The complete FQDN of your Active Directory domain:
 - **Example** : **contoso.local** or **it.contoso.local** ; this is based on where the gMSA is located.

- **Single-label domain** : Select this if your organization uses DNS names without a suffix.
- **Password** : This will be grayed out when we choose the *Group managed service account checkbox*.

SINGLE-LABEL DOMAIN

*It is uncommon for organizations to use single-label domains, or SLDs for short, which are DNS names without suffixes such as **.com** or **.org**. Although some Microsoft products can operate in SLD environments, it's not recommended for new deployments due to compatibility concerns with certain products and versions.*

Verifying the DSA

Now, let's verify the configuration of our DSA. Don't worry if this feels a bit advanced right now – we'll dive deeper into the different PowerShell functions available in the MDI PowerShell module in [Chapter 3](#). For now, follow along with the steps here, and rest assured that you'll gain a more comprehensive understanding as we progress through the book.

We need to verify that our DSA has the right configuration, and we can do that with the following command. With the `Identity` parameter, we will write the username of our DSA, and with the `Detailed` parameter, we will get more output about the validation status:

```
Test-MDIDSA -Identity "MDIGMSA$" -Detailed
```

If you have issues, you will get a Boolean value of `$false`; we then need to circle back to the configuration of that issue and see if we missed some of the settings necessary. If you get all `$true`, you are ready for the next step.

Defender XDR unified RBAC

It is now possible to have centralized permissions management for, at the time of writing this book, several of the Defender products. The **Defender XDR portal** is used for your daily security tasks, such as threat detection, threat protection, and response across all the Defender suite and in one single portal (we have all been waiting for a consolidated portal). It's very important that we give the right set of permissions to our users (not end users to be clear) so they can do their daily tasks, such as manage alerts, look at activities, change configuration and other system settings, or just have full view permission.

IMPORTANT NOTE

Some of the highly privileged Entra global roles, such as Global Administrator, will continue to have admin privileges, and therefore, the Defender XDR security products will respect these roles by maintaining their extensive access rights. This ensures that critical administrative capabilities (such as for emergency access/break-glass accounts) remain uninterrupted, facilitating a seamless transition and ongoing management within the unified RBAC framework.

First, we need to build our groups that will be linked to the custom roles that we are about to create. It's crucial to set the `isAssignableToRole` property to `true` when creating these groups. This property makes the groups role-assignable, allowing roles to be securely assigned to them. By doing this, we enhance security by preventing unauthorized privilege escalation. It also ensures that only designated administrators with the Privileged Role Administrator role in Microsoft Entra ID can create and manage these groups.

Additionally, you can delegate the management of these groups by assigning trusted group owners. Group owners can add or remove members from the group but should be carefully selected to prevent unauthorized access or privilege escalation.

This setup can absolutely be adjusted for your organization's needs, but here are some pointers and ideas for you.

To architect unified RBAC within MDI, you would typically define and create specific personnel roles in Microsoft Entra ID, which are then integrated with specific permissions in Defender XDR. Here's a basic framework:

1. **Role definition :** Define roles based on job functions within your organization that interact with MDI. Common roles might include the following:

- Security Administrator
- Security Analyst Tier 1 (Read-Only)
- Security Analyst Tier 2 (Limited Access)
- Security Analyst Tier 3 (Manage)

2. **Group mapping in Entra ID :** Create role-assignable groups in Microsoft Entra ID corresponding to these roles. These groups will have the `isAssignableToRole` property set to `true` , allowing them to be assigned roles. Follow your organization's naming conventions. Here's how to proceed:

A. **Creating role-assignable groups :**

- **Administrative role requirement :** You must be assigned at least the Privileged Role Administrator role to create role-assignable groups.
- **Set isAssignableToRole to true :** When creating each group, set the `isAssignableToRole` property to `true` . This property is immutable and cannot be changed after creation. Existing groups cannot have this property set retroactively.
- **Membership type must be Assigned :** The membership type for role-assignable groups must be `Assigned` . Dynamic groups are not permitted to prevent unintended privilege assignments through automated processes.
- **No group nesting :** Group nesting is not supported for role-assignable groups. A group cannot be added as a member of another role-assignable group to prevent indirect role assignments.

B. **Group examples :**

- **MDI-SecAdmin-Full** : This group is designated for Security Administrators with full access privileges.
- **MDI-SecAnalystT1-ReadOnly** : Tier 1 Security Analysts in this group have read-only access, suitable for those primarily in monitoring roles.
- **MDI-SecAnalystT2-Limited** : Tier 2 Security Analysts have limited access, providing them capabilities for certain operational tasks without full administrative rights.
- **MDI-SecAnalystT3-Manage** : Tier 3 Security Analysts in this group can manage configurations and settings within MDI, tailored for senior analysts responsible for more interactive or complex tasks.

3. **Assign permissions in Defender XDR** : Link these groups to specific permissions in Defender XDR that align with their responsibilities, such as these:

- **MDI-SecAdmin-Full** :
 - **Security operations** : All read and manage permissions
 - **Security posture** : All read and manage permissions
 - **Authorization and settings** : All read and manage permissions
- **MDI-SecAnalystT1-ReadOnly** :
 - **Security operations** : All read only permissions
 - **Security posture** : All read only permissions
 - **Authorization and settings** : None
- **MDI-SecAnalystT2-Limited** :
 - **Security operations (custom permissions)** : Alerts (manage), Response (manage), Basic live response (manage), File collection (manage)
 - **Security posture** : All read and manage permissions
 - **Authorization and settings (custom permission)** : Security settings > Select custom permission > Detect tuning (manage)
- **MDI-SecAnalystT3-Manage** :
 - **Security operations** : All read and manage permissions
 - **Security posture** : All read and manage permissions
 - **Authorization and settings** : Select Custom permission
 - **Security settings** : Detect tuning (manage), Core security settings (read), Core security settings (manage)
 - **System settings** : Read and manage permissions

4. **Unified RBAC configuration** : Configure Unified RBAC in Defender XDR to enforce these permissions automatically based on group membership from Entra ID.

5. **Continuous review and adjustment**: Regularly review and adjust the roles, groups, and permissions to ensure they meet evolving security needs and organizational changes.

FIRST TIME ACCESSING MDI

To be able to create the MDI workspace for the first time in your tenant, you need the Global Administrator or Security Administrator role assigned to your account.

As we bring our post-installation activities to a close, it's important to take a moment and appreciate the groundwork we've laid for a secure and smooth-running MDI environment. We've covered everything from getting our DSA configured just right to setting up SAM-R and even fine-tuning our gMSA settings in Defender XDR. Each of these steps isn't just a box to tick – they're crucial for making sure everything runs like a well-oiled machine. As you may have noticed from earlier experience in the field, many default settings when we just rush through the installation wizard makes the product or solution work – but aren't configured in the most secure way. That's why it is so important to understand everything around a specific solution or product.

And let's not forget the cherry on top – configuring RBAC in Defender XDR, which ensures that only the right people have the keys to the castle. By following these steps, you've done more than just setting up MDI; you've built a strong defense system that's ready to take on whatever comes its way.

Now, let's look back at what we've accomplished in this chapter as we move to the summary section.

Summary

In this chapter, we explored the expanding threat landscape and introduced the concept of ITDR.

We delved into the pivotal role of MDI in safeguarding crucial assets, particularly identities, and detailed steps for planning and implementing MDI using a comprehensive checklist.

We also examined the importance of secure communication and considered how proxy solutions might enhance MDI deployment security.

Furthermore, we discussed why gMSAs are currently the safest method for MDI sensors to detect threats.

As we move forward, the next chapter will focus on managing MDI and automating tasks using PowerShell and the MDI PowerShell module, building on the insights and best practices gathered in this chapter.

3

Leveraging MDI PowerShell for Automation and Management

In this chapter, we will explore the synergy between **Microsoft Defender for Identity (MDI)** and PowerShell, unveiling how this powerful scripting tool can automate and refine the management of MDI. This chapter is designed to equip you with the knowledge to harness PowerShell's scripting capabilities for robust security management, transforming complex tasks into efficient and automated processes. This chapter is not just about understanding commands; it's about mastering the art of automation to transform reactive security measures into proactive shields.

By integrating MDI with Azure services, such as **Azure Monitor Agent (AMA)** and **Microsoft Sentinel**, we'll explore advanced strategies for monitoring and investigating system configurations, ensuring a proactive approach to cybersecurity within your organization. Through hands-on examples and insightful explanations, you will learn to navigate the complexities of MDI reports and leverage them to identify misconfigurations that could be symptomatic of deeper security issues.

After reading this chapter, you should get a good understanding of how the MDI PowerShell module works and how you can use these two components in automation capabilities.

In this chapter, we will cover the following:

- Primer on the MDI PowerShell module
- Crafting advanced PowerShell scripts for MDI management
- Automation in action – case studies and scripting scenarios

Let's get started!

Technical requirements

The completion of [Chapter 2](#) is a prerequisite for this section, and as a reminder, you also require the following:

- Microsoft tenant
- A Microsoft subscription that includes MDI, such as Microsoft 365 E5 or Microsoft 365 E3 + E5 Security
- Basic Microsoft 365 knowledge
- Active Directory knowledge
- A virtual or physical server environment with Active Directory installed and configured:
 - **Optional : Active Directory Federation Services (AD FS), Active Directory Certificate Services (AD CS) and Entra Connect** installed and configured

- Basic PowerShell knowledge :
 - Windows PowerShell 5.1 or PowerShell 7.4 or later
 - Basic networking knowledge

All the code examples for this chapter can be found on GitHub at

<https://github.com/PacktPublishing/Microsoft-Defender-for-Identity-in-Depth/tree/main/Chapter03>

Primer on the MDI PowerShell module

The second chapter and the overall documentation for Defender for Identity provide you with instructions for setting up the necessary configurations by hand. Yet, this manual process may lead to mistakes, consume considerable time, and present challenges in managing within complicated, multi-domain settings. For instance, implementing audit checks on read property activities can lead to an excessive burden on domain controllers. Additionally, if the correct events are not recorded, MDI might miss certain activities, and I bet those activities are very necessary for your SecOps team. It's important to follow the setup guidelines carefully to ensure the system operates effectively and efficiently.

The Defender for Identity PowerShell module is here to help us with the automation for configuring domain controllers and other sensor servers.

Installing the MDI PowerShell module

Installing the MDI PowerShell module, known as `DefenderForIdentity`, is a critical step in enhancing your sensor server's security. The module requires Windows PowerShell 5.1 or PowerShell 7.4 or later.

For systems with internet access, the module can be installed directly from the PowerShell Gallery using the `Install-Module -Name DefenderForIdentity` command. For environments without direct internet access, the module can be downloaded from a machine with internet access and then transferred to the target server using a secure method. Once on the target server, the module is imported into PowerShell with the `Import-Module -Name DefenderForIdentity` command.

In early 2022, Microsoft updated its guidance regarding internet access for domain controllers. The new recommendation now points to the fact that while domain controllers should *not* have unrestricted internet access, and browsing the internet from these servers continues to be off-limits, a total ban on internet access is no longer advised. I believe that you agree that we must emphasize a comprehensive security strategy, which includes advanced threat detection systems, and that it is

essential to effectively manage, monitor, and protect against potential breaches, rather than relying on the outdated perception that domain controllers will remain uncompromised if isolated from the internet. This nuanced approach aims to balance security with functionality in today's complex digital environment.

Remember, the following process requires careful attention to detail and understanding of PowerShell's module system. Always test in a controlled environment before deploying to production servers.

Installing the module on sensor servers with internet access

To install the module, you need to follow these steps:

1. On the server where you want to install the module, start PowerShell and type the following:

```
Install-Module DefenderForIdentity
```

2. To verify that the module is installed correctly, type the following:

```
Get-Module -ListAvailable -Name DefenderForIdentity
```

Importing the module on sensor servers without internet access

To install and import the module on a server without internet access, you will need to perform a manual download and installation. Here's one way to do it:

1. On an internet-connected machine, start PowerShell and type the following:

```
Save-Module -Name DefenderForIdentity -Path C:\temp\ -Verbose
```

- The Save-Module cmdlet is from the PowerShellGet module that will download the necessary module and its dependencies. Please modify the Path parameter to reference a different folder and make sure the folder exists. The Verbose parameter outputs detailed progress to the console, helping you monitor each step and quickly spot any errors for easier troubleshooting.

2. Once downloaded, transfer the module folder to the target server using a secure method.

3. On the target server, you'll need to decide whether to install the module for all users or just the current user:

- I. For all users, do the following:

- For Windows PowerShell 5.1, place it in the following:

```
$env:ProgramFiles\WindowsPowerShell\Modules
```

- For PowerShell 7, place it in the following:

```
$env:ProgramFiles\PowerShell\Modules
```

- II. For the current user, do the following:

- For Windows PowerShell 5.1, place it in the following:

```
$env:USERPROFILE\Documents\WindowsPowerShell\Modules
```

- For PowerShell 7, place it in the following:

```
$env:USERPROFILE\Documents\PowerShell\Modules
```

4. Ensure that the module path is included in the **PSModulePath** environment variable. If it's not, you can add it by modifying the variable or using the **Import-Module** cmdlet with the full path to the module.

If you have installed PowerShell 7 on your server, you can check the module paths of each PowerShell version.

In Windows PowerShell 5.1:

```
C:\Users\<user>\Documents\WindowsPowerShell\Modules  
C:\Program Files\WindowsPowerShell\Modules  
C:\WINDOWS\System32\WindowsPowerShell\v1.0\Modules
```

In PowerShell 7:

```
C:\Users\<user>\Documents\PowerShell\Modules  
C:\Program Files\PowerShell\Modules  
C:\Program Files\PowerShell\7\Modules  
C:\Program Files\WindowsPowerShell\Modules  
C:\WINDOWS\System32\WindowsPowerShell\v1.0\Modules
```

- To verify and add a path to the PSModulePath environment variable, type the following:

```
$env:PSModulePath -split ';'  
$env:PSModulePath = $env:PSModulePath + ";C:\your\newModulePath"
```

5. Use **Import-Module DefenderforIdentity** to import the module into your PowerShell session. If the module is not recognized, verify the path, and ensure that the execution policy allows the script to run.

Module file overview

The files we get after we have installed the module are as follows:

- **DefenderForIdentity-help.xml** :
 - An XML file offering detailed help and instructions for utilizing the MDI PowerShell commands, designed to assist users in navigating the module's features
- **DefenderForIdentity.psd1** :
 - The **.psd1** file acts as the module manifest, cataloging the module's contents and providing instructions for its execution. It is formatted as a text file containing a hash table with key-value pairings, essential for defining the module's structure and usage.
- **DefenderForIdentity.psm1** :
 - This **.psm1** file contains the PowerShell script module, encompassing crucial scripts and commands for MDI management. It facilitates direct interaction and administration of MDI functionalities within PowerShell.

- **LICENSE.txt :**
 - The license file specifies the MIT (Massachusetts Institute of Technology) License governing the module, an open source license granting extensive freedoms to use, modify, and distribute the software. The sole requirement is that the original license and copyright notices are maintained in any major redistributions of the software.

Understanding the module and its functions

If we open the `DefenderForIdentity.psm1` file, you will see everything, including what types of other modules it requires under the `#requires` statement, all the different settings, and all the helper functions:

```
#requires -Version 4.0
#requires -Modules ActiveDirectory, GroupPolicy
```

The external module dependencies that we need to have installed are the `ActiveDirectory` and `GroupPolicy` modules. As you can see, it requires a minimum of version 4 of Windows PowerShell, which came with Windows Server 2012 R2 – and these servers should not be set up as a Tier 0 system, such as domain controllers, certificate servers, or federation servers, as this operating system version has reached its end of life and is no longer supported. Yes, you can buy **Extended Security Updates (ESUs)** but no, don't use old end-of-life operating systems as your most crucial workload.

For the helper functions listed in the `FunctionsToExport` field, this section of the PowerShell module manifest specifies which functions are visible and usable when the module is imported. By including only selected helper functions in this list, the module ensures that you can access and utilize specific functionalities, enhancing both the usability and security of the module. The functions that will be enabled in your PowerShell session are as follows:

- `*- MDIConfiguration`
- `New-MDIDSA`
- `Test-MDIDSA`
- `Test-MDISensorApiConnection`
- `*- MDISensorProxyConfiguration`
- `New-MDIClusterReport`

The most important part here is that we need to test our configuration so that we have the required and recommended configuration applied. The `Test-MDIClusterReport` function will start the function called `Get-MDIClusterReport`, which is a few rows larger if you look at the `DefenderForIdentity.psm1` file, and this function is looking at the different configurations listed here:

- **AdfsAuditing** : This will check whether necessary permissions are set on the auditing entry for the AD FS container under `CN=ADFS,CN=Microsoft,CN=Program Data`.

- **AdvancedAuditPolicyCAs** : If you have AD CS servers, configuring the recommended Audit Certification Services settings in a Group Policy is crucial to detect if someone is trying to exploit a misconfiguration in AD CS.
- **AdvancedAuditPolicyDCs** : This will check the recommended advanced audit settings for domain controllers, such as auditing the **Computer Account Management and Security Group Management** settings.
- **CAAuditing** : This will see if the CA (Certificate Authority) auditing is set on the CA servers, which can be done in various ways, such as enabling auditing on the properties of the CA server or by using the following command:

```
certutil -setreg CA\AuditFilter 127
```

- **ConfigurationContainerAuditing** : This check will see whether you have set the correct **system access control list (SACL)** in the configuration container within Active Directory. Specifically, it verifies that the **Write all properties** permission is applied to **This object and all descendant objects**.
- **DomainObjectAuditing** : As the configuration container, we are also trying to see whether the correct SACL is set on the domain level. With this audit setting, we will collect event 4662. This event is generated every time an operation is performed on an Active Directory object.
- **NTLMAuditing** : In this check, we will see whether the correct **New Technology LAN Manager (NTLM)** auditing settings are applied, such as restricting sending NTLM traffic, auditing receiving NTLM traffic, and auditing NTLM authentication in the domain. NTLM originates from **Windows NT** and is still used in some scenarios but should be avoided due to its severe vulnerability.
- **ProcessorPerformance** : It is very important not just for domain controllers but for apps and servers that are sensitive to processor performance changes to have the **High Performance** power scheme activated. This is also stated in the prerequisites of MDI. You can test and run the following code in PowerShell to see which power scheme you currently have activated:

```
& "$env:SystemRoot)\system32\powervcfg.exe" @(' /GETACTIVESCHEME')
```

- **All** : This will take all the preceding tests and run them at once.

Since version 1.0.0.1 of the MDI PowerShell module, we can create the gMSA account with ease. The following PowerShell command will create the gMSA account and the group specified in the **GmsaGroupName** parameter. The cmdlet will also add both the new group name as well as the domain controllers group to **PrincipalsAllowedToRetrieveManagedPassword**, meaning that members of those groups can see the password of the gMSA account:

```
New-MDIDSA -Identity MDIGMSA -GmsaGroupName MDIGroup
```

You will get **True** or **False** if the account and group can be created in Active Directory. To view whether or not the groups are added to the gMSA account, run the coming PowerShell code:

```
Get-ADServiceAccount -Identity MDIGMSA -Properties PrincipalsAllowedToRetrieveManagedPassword
```

You can also use the **Test-MDIDSA** cmdlet from the **DefenderForIdentity** module to verify that the account has all the requirements:

```
Test-MDIDSA -Identity MDIGMSA -Detailed
```

Make sure that you get the value **True** on all of the tests.

To test all configurations, on the machine you have installed the module on (which could be domain controller, etc.), you can run the following code in PowerShell:

```
Test-MDIConfiguration -Mode LocalMachine -Configuration All -Verbose
```

For the `Mode` parameter, you can choose between `LocalMachine` OR `Domain`.

The output will be the test result of each configuration (see *Figure 3.1*). You will now see the verbose output when we run the `Get-MDIConfiguration` function:

```
PS C:\> Get-MDIConfiguration -Mode Domain -Configuration All -Verbose
VERBOSE: Validating ADFS container auditing
VERBOSE: Test passed
VERBOSE: Validating GPO: Microsoft Defender for Identity - Advanced Audit Policy for Cas
VERBOSE: 'Microsoft Defender for Identity - Advanced Audit Policy for CAs' - GPO not found
VERBOSE: Test failed
VERBOSE: Validating GPO: Microsoft Defender for Identity - Advanced Audit Policy for DCs
VERBOSE: 'Microsoft Defender for Identity - Advanced Audit Policy for DCs' - GPO not found
VERBOSE: Test failed
VERBOSE: Validating GPO: Microsoft Defender for Identity - Auditing for Cas
VERBOSE: 'Microsoft Defender for Identity - Auditing for CAs' - GPO not found
VERBOSE: Test failed
VERBOSE: Validating Exchange related configuration container auditing
VERBOSE: Test passed
VERBOSE: Validating Domain Object auditing
VERBOSE: Test failed
VERBOSE: Validating GPO: Microsoft Defender for Identity - NTLM Auditing for DCs
VERBOSE: 'Microsoft Defender for Identity - NTLM Auditing for DCs' - GPO not found
VERBOSE: Test failed
VERBOSE: Validating GPO: Microsoft Defender for Identity - Processor Performance
VERBOSE: 'Microsoft Defender for Identity - Processor Performance' - GPO not found
VERBOSE: Test failed
```

In the following figure, you will see the verbose output and the result of the `Get-MDIConfiguration` run.

```

PS C:\> Get-MDIConfiguration -Mode Domain -Configuration All -verbose
VERBOSE: Validating ADFS container auditing
VERBOSE: Test passed
VERBOSE: Validating GPO: Microsoft Defender for Identity - Advanced Audit Policy for CAS
VERBOSE: 'Microsoft Defender for Identity - Advanced Audit Policy for CAS' - GPO not found
VERBOSE: Test failed
VERBOSE: Validating GPO: Microsoft Defender for Identity - Advanced Audit Policy for DCs
VERBOSE: 'Microsoft Defender for Identity - Advanced Audit Policy for DCs' - GPO not found
VERBOSE: Test failed
VERBOSE: Validating GPO: Microsoft Defender for Identity - Auditing for CAS
VERBOSE: 'Microsoft Defender for Identity - Auditing for CAS' - GPO not found
VERBOSE: Test failed
VERBOSE: Validating Exchange related configuration container auditing
VERBOSE: Test passed
VERBOSE: Validating Domain Object auditing
VERBOSE: Test failed
VERBOSE: Validating GPO: Microsoft Defender for Identity - NTLM Auditing for DCs
VERBOSE: 'Microsoft Defender for Identity - NTLM Auditing for DCs' - GPO not found
VERBOSE: Test failed
VERBOSE: Validating GPO: Microsoft Defender for Identity - Processor Performance
VERBOSE: 'Microsoft Defender for Identity - Processor Performance' - GPO not found
VERBOSE: Test failed

Configuration          Mode      Status Details
-----          ----      ---- -----
CAAuditing           Domain    False   'Microsoft Defender for Identity - Auditing for CAS' - GPO not found
NTLMAuditing          Domain    False   'Microsoft Defender for Identity - NTLM Auditing for DCs' - GPO not found
ConfigurationContainerAuditing Domain    True    Microsoft Exchange Services container not found
DomainObjectAuditing  Domain    False   {@{Account=Everyone; SecurityIdentifier=S-1-1-0; AccessMask=786464; AccessMa...
AdfsAuditing          Domain    True    Microsoft ADFS container not found
AdvancedAuditPolicyDCS Domain    False   'Microsoft Defender for Identity - Advanced Audit Policy for DCs' - GPO not ...
ProcessorPerformance  Domain    False   'Microsoft Defender for Identity - Processor Performance' - GPO not found
AdvancedAuditPolicyCAS Domain    False   'Microsoft Defender for Identity - Advanced Audit Policy for CAS' - GPO not ...

```

Figure 3.1 – Verbose output from Get-MDIConfiguration Domain

The verbose output indicates that some configuration is needed before the MDI sensor is fully operational, which should help ensure we are not caught off guard.

The output from the `LocalMachine` value in the `Mode` parameter will show some difference because it will see whether you are running the commands on another server, such as the AD CS server (see *Figure 3.2*).

```

PS C:\> Get-MDIConfiguration -Mode LocalMachine -Configuration All -verbose
VERBOSE: Validating ADFS container auditing
VERBOSE: Test passed
VERBOSE: Validating Advanced Audit Policy for Certificate Authority servers
VERBOSE: CertSvc service not found. This is not a Certificate Authority server
VERBOSE: Test passed
VERBOSE: Validating Advanced Audit Policy for Domain Controllers
VERBOSE: Test failed
VERBOSE: Validating Certificate Authority server auditing settings
VERBOSE: CertSvc service not found. This is not a Certificate Authority server
VERBOSE: Test passed
VERBOSE: Validating Exchange related configuration container auditing
VERBOSE: Test passed
VERBOSE: Validating Domain Object auditing
VERBOSE: Test failed
VERBOSE: Validating NTLM auditing
VERBOSE: Test failed
VERBOSE: Validating Processor Performance
VERBOSE: Test failed

Configuration          Mode      Status Details
-----          ----      ---- -----
CAAuditing           LocalMachine True    CertSvc service not found. This is not a Certificate Authority server
NTLMAuditing          LocalMachine False   {@{Path=HKLM:(System\CurrentControlSet\services\Netlogon\Parameters}; ...
ConfigurationContainerAuditing LocalMachine True    Microsoft Exchange Services container not found
DomainObjectAuditing  LocalMachine False   {@{Account=Everyone; SecurityIdentifier=S-1-1-0; AccessMask=786464; Ac...
AdfsAuditing          LocalMachine True    Microsoft ADFS container not found
AdvancedAuditPolicyDCS LocalMachine False   {@{Machine Name=DC01; Policy Target=system; Subcategory=Security syste...
ProcessorPerformance  LocalMachine False   Power Scheme GUID: 381b4222-f694-41f0-9685-ff5bb260df2e (Balanced)
AdvancedAuditPolicyCAS LocalMachine True    CertSvc service not found. This is not a Certificate Authority server

PS C:\>

```

Figure 3.2 – Verbose output from Get-MDIConfiguration LocalMachine

To view the configuration in an easier-to-read HTML format, you can use the `New-MDIConfigurationReport` function. To execute this command successfully, you must provide the

mandatory parameter `Path`, which allows you to save both the HTML and JSON files in the location you specify:

```
New-MDIConfigurationReport -Path C:\temp\MDI_Reports
```

The report can then be viewed in your web browser (see *Figure 3.3*).

The MDI configuration report validates the domain SACLs and the presence of related configuration

Configuration	Status	Command to fix
AdfsAuditing	Passed	Set-MDIConfiguration -Mode Domain -Configuration AdfsAuditing
AdvancedAuditPolicyCAs	Failed	Set-MDIConfiguration -Mode Domain -Configuration AdvancedAuditPolicyCAs
AdvancedAuditPolicyDCs	Failed	Set-MDIConfiguration -Mode Domain -Configuration AdvancedAuditPolicyDCs
CAAuditing	Failed	Set-MDIConfiguration -Mode Domain -Configuration CAAuditing
ConfigurationContainerAuditing	Passed	Set-MDIConfiguration -Mode Domain -Configuration ConfigurationContainerAuditing
DomainObjectAuditing	Failed	Set-MDIConfiguration -Mode Domain -Configuration DomainObjectAuditing
NTLMAuditing	Failed	Set-MDIConfiguration -Mode Domain -Configuration NTLMAuditing
ProcessorPerformance	Failed	Set-MDIConfiguration -Mode Domain -Configuration ProcessorPerformance

Note: After creating the Group Policy Objects, it may take up to 120 minutes for the settings to apply

Full details file can be found at C:\temp\MDI_Reports\MDI-configuration-report-CONTOSO.LOCAL.json

Created by the [DefenderForIdentity](#) PowerShell module

Figure 3.3 – MDI configuration report

For the JSON file, you can process it in PowerShell by using the `Get-Content` cmdlet with the specified file path, then pipe the output into the `ConvertFrom-Json` cmdlet to parse the JSON-formatted strings and convert them into PowerShell objects. With PowerShell objects, you can easily manipulate and manage the data within your scripts, allowing for automated processing, analysis, and reporting.

If you try the following command, you will get the same type of output as the `Get-MDIConfiguration` command, but since we are now working with PowerShell objects, let's try to put the output in a list instead of a table:

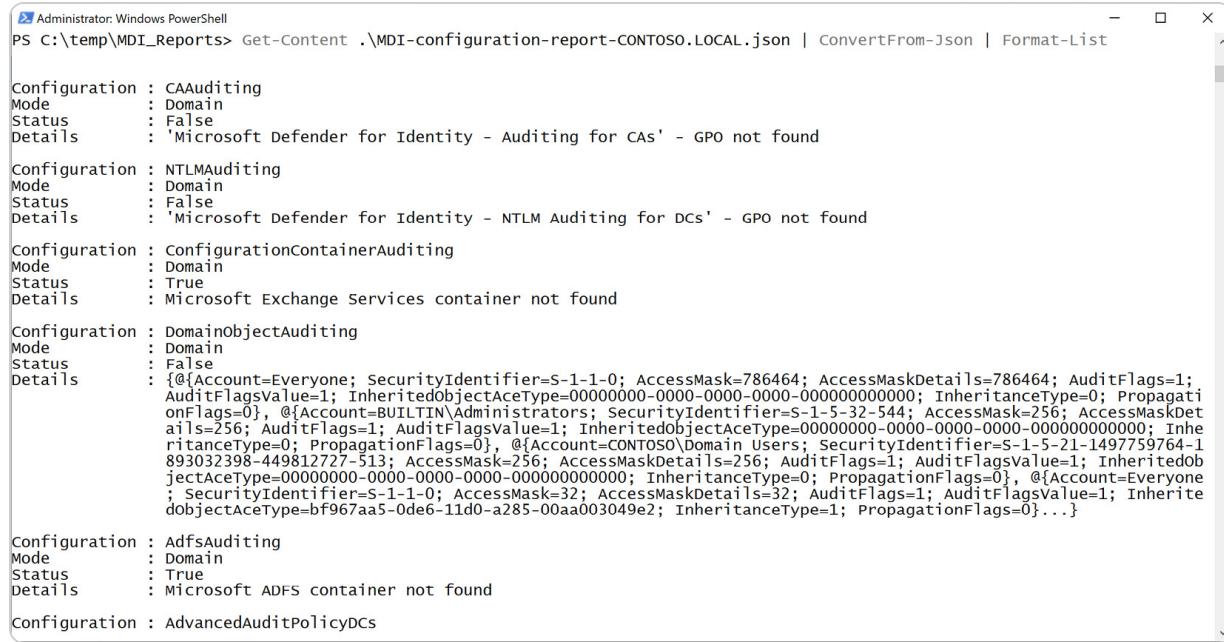
```
Get-Content -Path C:\temp\MDI_Reports\MDI-configuration-report-CONTOSO.LOCAL.json |  
ConvertFrom-Json
```

If we pipe the output from the `ConvertFrom-Json` cmdlet to `Format-List`, you will see the entire output and not be cut off because of the PowerShell window size:

```
Get-Content -Path C:\temp\MDI_Reports\MDI-configuration-report-CONTOSO.LOCAL.json |
```

```
ConvertFrom-Json | Format-List
```

As you can see, it's easier to view the result in PowerShell.



```
Administrator: Windows PowerShell
PS C:\temp\MDI_Reports> Get-Content .\MDI-configuration-report-CONTOSO.LOCAL.json | ConvertFrom-Json | Format-List

Configuration : CAAuditing
Mode         : Domain
Status       : False
Details      : 'Microsoft Defender for Identity - Auditing for CAs' - GPO not found

Configuration : NTLMAuditing
Mode         : Domain
Status       : False
Details      : 'Microsoft Defender for Identity - NTLM Auditing for DCs' - GPO not found

Configuration : ConfigurationContainerAuditing
Mode         : Domain
Status       : True
Details      : Microsoft Exchange Services container not found

Configuration : DomainObjectAuditing
Mode         : Domain
Status       : False
Details      : {@{Account=Everyone; SecurityIdentifier=S-1-1-0; AccessMask=786464; AccessMaskDetails=786464; AuditFlags=1; AuditFlagsValue=1; InheritedObjectAceType=00000000-0000-0000-000000000000; InheritanceType=0; PropagationFlags=0}, @{Account=BUILTIN\Administrators; SecurityIdentifier=S-1-5-32-544; AccessMask=256; AccessMaskDetails=256; AuditFlags=1; AuditFlagsValue=1; InheritedObjectAceType=00000000-0000-0000-0000-000000000000; InheritanceType=0; PropagationFlags=0}, @{Account=CONTOSO\Domain Users; SecurityIdentifier=S-1-5-21-1497759764-1893032398-449812727-513; AccessMask=256; AccessMaskDetails=256; AuditFlags=1; AuditFlagsValue=1; InheritedObjectAceType=00000000-0000-0000-000000000000; InheritanceType=0; PropagationFlags=0}, @{Account=Everyone; SecurityIdentifier=S-1-1-0; AccessMask=32; AccessMaskDetails=32; AuditFlags=1; AuditFlagsValue=1; InheritedObjectAceType=bf967aa5-0de6-11d0-a285-00aa003049e2; InheritanceType=1; PropagationFlags=0}...}

Configuration : AdfsAuditing
Mode         : Domain
Status       : True
Details      : Microsoft ADFS container not found

Configuration : AdvancedAuditPolicyDCS
```

Figure 3.4 – Listed output

To play around with the PowerShell object, make sure to save the output in a variable to be able to work with it easier. If we save the output to the `$report` variable, we then can dig into each of the configuration tests, such as `DomainObjectAuditing`, which has a lot of information under the `Details` key:

```
$report = Get-Content -Path C:\temp\MDI_Reports\MDI-configuration-report-CONTOSO.LOCAL.json | ConvertFrom-Json
$report[3].Details
```

In the following figure, you will see the report variable, which, in this case, is the `DomainObjectAuditing` test and its status and other details.

```

Administrator: Windows PowerShell
PS C:\temp\MDI_Reports> $report[3]
Configuration      Mode    Status Details
-----  -----
DomainObjectAuditing Domain  False  {@{Account=Everyone; SecurityIdentifier=S-1-1-0; AccessMask=786464; AccessMaskDetails=...}

PS C:\temp\MDI_Reports> $report[3].Details

Account          : Everyone
SecurityIdentifier : S-1-1-0
AccessMask        : 786464
AccessMaskDetails : 786464
AuditFlags       : 1
AuditFlagsValue  : 1
InheritedObjectAceType : 00000000-0000-0000-0000-000000000000
InheritanceType   : 0
PropagationFlags  : 0

Account          : BUILTIN\Administrators
SecurityIdentifier : S-1-5-32-544
AccessMask        : 256
AccessMaskDetails : 256
AuditFlags       : 1
AuditFlagsValue  : 1
InheritedObjectAceType : 00000000-0000-0000-0000-000000000000
InheritanceType   : 0
PropagationFlags  : 0

Account          : CONTOSO\Domain Users
SecurityIdentifier : S-1-5-21-1497759764-1893032398-449812727-513
AccessMask        : 256
AccessMaskDetails : 256
AuditFlags       : 1
AuditFlagsValue  : 1
InheritedObjectAceType : 00000000-0000-0000-0000-000000000000

```

Figure 3.5 – PowerShell object

Before you install the MDI sensor, you could also go through the necessary configuration with PowerShell and the `Get-MDIConfiguration` function within the module. As in the previous example, we see the `Details` property with information on what needs to be configured. We can run the following command on a server, and in this example, we will look at `LocalMachine` for the `ProcessorPerformance` setting and see which power setting we currently have:

```
Get-MDIConfiguration -Mode LocalMachine -Configuration ProcessorPerformance | Select-Object -ExpandProperty Details
```

Now that we have investigated how the module is structured and how we can install it, have played around with some of the functions, and the first checks are done, we are ready to begin utilizing it.

Crafting advanced PowerShell scripts for MDI management

As you may have understood by now, it is crucial to have the right advanced auditing settings in place, the correct GPO (Group Policy Objects) created and linked to our servers, which will act as sensor servers, and other settings in place, otherwise, we will have blind spots in our detection capability.

It is one thing to install and configure products on servers or other endpoints, but in this case, we need to carefully monitor the baseline configuration so we can be alerted and notified if the configuration changes. At the time of writing this book, we do have a page of current health issues in the **Identity** blade within the Microsoft Defender XDR portal.

If we look at the page, we can see health issues on the sensor (see *Figure 3.6*).

Health Issues

The Microsoft Defender for Identity Health Center lets you know when there's a problem with your Defender for Identity instance, by raising a health alert. [Learn more](#)

Global health issues (2) Sensor health issues (2)

Export 2 items Search All

Filter Reset Filters

Status: Open Issue: Any Severity: Any Sensor Name: Any

Issue	Severity	Status	Generation time	Description	Sensor Name
Sensor with non-optimal power settings	Low	Open	Mar 3, 2024 12:18 PM	The operating system's power mode o...	DC01.contoso.lo...
NTLM Auditing is not enabled	Medium	Open	Mar 3, 2024 12:18 PM	NTLM Auditing is not enabled on DC01...	DC01.contoso.lo...

Figure 3.6 – Sensor health issues

You can also see global health issues (see *Figure 3.7*), as advanced auditing is not configured correctly.

Health Issues

The Microsoft Defender for Identity Health Center lets you know when there's a problem with your Defender for Identity instance, by raising a health alert. [Learn more](#)

Global health issues (5) Sensor health issues (4)

Export 5 items Search All

Filter Reset Filters

Status: Open Issue: Any Severity: Any

Issue	Severity	Status	Generation time	Description
Auditing on the ADFS container is not...	Medium	Open	Jan 9, 2024 2:43 PM	Auditing on the ADFS container is not...
Directory Services Advanced Auditing...	Medium	Open	Jan 9, 2024 2:43 PM	Directory Services Advanced Auditing ...
Auditing on the Configuration contain...	Medium	Open	Jan 9, 2024 2:43 PM	Auditing on the Configuration contain...
Directory Services Object Auditing is ...	Medium	Open	Jan 9, 2024 2:43 PM	Directory Services Object Auditing is n...

Figure 3.7 – Global health issues

We can be alerted through email if we get these health issues, but that may not be the correct way for your organization. You may want to get these alerts ingested into Azure Monitor and Log Analytics, ticketing systems, such as ServiceNow, Teams adaptive card notifications, or even into Azure DevOps

as a task – the possibility and use cases can be truly expanded in so many areas. We will go through some of them later.

Next up, we will look at Microsoft Graph REST API, where we can get health issues in a programmatical way instead of relying on email from the service when we get some type of issue.

Health issues API

Microsoft Graph acts as a centralized gateway to data and insights across Microsoft 365, delivering a streamlined API framework for accessing a wide range of information and intelligence from Microsoft 365, Enterprise Mobility + Security, and Windows. With a single endpoint at <https://graph.microsoft.com>, Microsoft Graph exposes APIs from numerous Microsoft cloud services. This consolidation facilitates the efficient management of API requests and simplifies the development process for applications that interact with a variety of data sources.

Within Microsoft Graph REST API and under the **Security** endpoint, we can now find the **Identity** library. The Defender for Identity health issues API enables the monitoring of the health status of your sensors and agents in your hybrid identity infrastructure. This API provides details on current health issues affecting your sensors, including issue type, status, configuration, and severity.

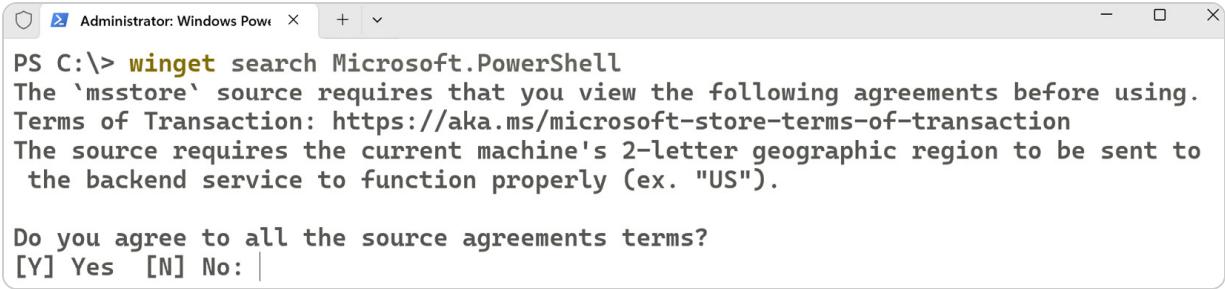
In this section, we will go through how we can talk to the health issues API through Microsoft Graph PowerShell SDK (software development kit). However, first, we need to install the `microsoft.Graph.Beta` module, which will include the security module,

`microsoft.Graph.Beta.Security`. At the time of writing this book, the only way to communicate with the health issues API is on the beta endpoint, and since Microsoft Graph APIs in the beta endpoint are prone to breaking changes, avoid using them in production apps. Instead, wait for a **general availability (GA)** release of an SDK that accesses the stable Microsoft Graph API endpoint.

Before we jump into the world of Microsoft Graph API, make sure to install PowerShell 7.x through the recommended way – `winget` :

```
winget search Microsoft.PowerShell
winget install --id Microsoft.PowerShell --source winget
```

If you haven't run `winget search` before, you need to agree to the source agreement terms (see *Figure 3.8*).

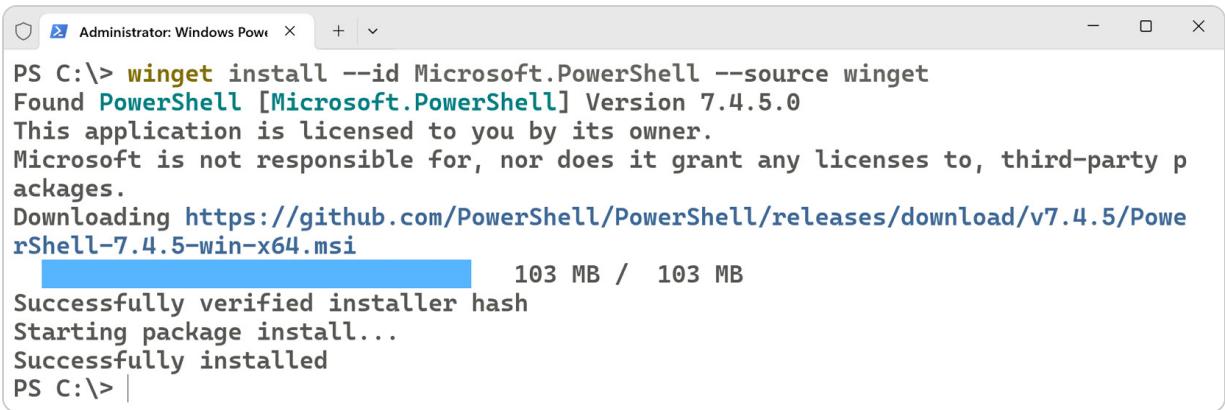


```
PS C:\> winget search Microsoft.PowerShell
The 'msstore' source requires that you view the following agreements before using.
Terms of Transaction: https://aka.ms/microsoft-store-terms-of-transaction
The source requires the current machine's 2-letter geographic region to be sent to
the backend service to function properly (ex. "US").

Do you agree to all the source agreements terms?
[Y] Yes [N] No: |
```

Figure 3.8 – winget agreement

Now, we can install PowerShell 7, and at the time of writing this book, we have version 7.4 (see *Figure 3.9*).



```
PS C:\> winget install --id Microsoft.PowerShell --source winget
Found PowerShell [Microsoft.PowerShell] Version 7.4.5.0
This application is licensed to you by its owner.
Microsoft is not responsible for, nor does it grant any licenses to, third-party p
ackages.
Downloading https://github.com/PowerShell/PowerShell/releases/download/v7.4.5/Powe
rShell-7.4.5-win-x64.msi
[██████████] 103 MB / 103 MB
Successfully verified installer hash
Starting package install...
Successfully installed
PS C:\> |
```

Figure 3.9 – winget install PowerShell 7

Close the old Windows PowerShell window and start the new PowerShell 7.

Now, let's start with the installation of some Microsoft Graph modules, importing the modules, authenticating to the Graph API, creating the app registration, creating the client secret to the app registration, and adding the permission called `SecurityIdentitiesHealth.Read.All` to the app.

Here is the PowerShell script we will be typing to be able to complete this task:

```
Install-Module Microsoft.Graph.Beta
if (-not (Get-Module -Name Microsoft.Graph.Authentication -ListAvailable)) {
    Install-Module -Name Microsoft.Graph.Authentication -Force
}
Import-Module -Name Microsoft.Graph.Authentication
Connect-MgGraph -Scopes AppRoleAssignment.ReadWrite.All,Application.ReadWrite.All -
NoWelcome
Import-Module -Name Microsoft.Graph.Beta.Security
$tenantId = (Get-MgContext).TenantId
$displayName = "MDI-Health-Issues-App-Registration"
$apiPermission = "SecurityIdentitiesHealth.Read.All"
$appRoleId = "f8dc971-5d83-4ele-aa95-ef44611ad351"
$appRegistration = New-MgApplication -DisplayName $displayName
$params = @{
    RequiredResourceAccess = @(
        @{
            ResourceAppId = "00000003-0000-0000-c000-000000000000"
```

```

    ResourceAccess = @(
        @{
            Id = $AppRoleId
            Type = "Role"
        }
    )
}

Update-MgApplication -ApplicationId $appRegistration.Id -BodyParameter $params
$secret = Add-MgApplicationPassword -ApplicationId $appRegistration.id
$graphSpId = $(Get-MgServicePrincipal -Filter "appId eq '00000003-0000-0000-c000-000000000000'").Id
$sp = New-MgServicePrincipal -AppId $appRegistration.appId
New-MgServicePrincipalAppRoleAssignment -ServicePrincipalId $sp.Id -PrincipalId $sp.Id -AppRoleId $AppRoleId -ResourceId $graphSpId

```

In this step, you need to remember the client secret we will be retrieving from the app registration. You will also be asked to do the following:

1. Press **Y** and then press the *Enter* key to be able to continue with the download and installation of the module from **PSGallery**; we will trust the repository.
2. Insert the client secret from the app registration (see *Figure 3.10*).

The script will do the following:

- Install the **Microsoft.Graph.Beta** module.
- If the module named **Microsoft.Graph.Authentication** is not installed, it will install it and then we will import the module.
- We need the appropriate account with appropriate permissions to connect to the Graph API and register the application. We will with the **Connect-MgGraph** sign in to Microsoft Graph with the permission scope of **AppRoleAssignment.ReadWrite.All** and **Application.ReadWrite.All**.
- In the **\$displayName** variable, enter the naming convention of your app registrations; in this example, I have typed in **MDI-Health-Issues-App-Registration**.
- Keep the other variables as they are unique, such as how **\$AppRoleId** is the GUID (globally unique identifier) for the **SecurityIdentitiesHealth.Read.All** application permission.
- **Update-MgApplication** needs a **BodyParameter** request, so we are creating that with the **\$param** parameter with all of the values required. The type of **resourceAccess** will be **"Role"**, which states that the API permission is for an application and not a delegated permission:
 - **Application permission : type = " Role"**
 - **Delegated permission : type = " Scope"**
- The secret (or password, if you will) gets generated and saved in the **\$ secret** variable.
- The last three lines will create an Enterprise application in Entra ID and grant the app registration to the tenant. (Did you know that the Enterprise application is the service principal in Entra ID?)

Now that we have completed the app registration, you can double-check to see whether it exists in the Entra ID portal under the **App Registration** blade, that it has the permission we want, and that the

secret is created. When you are in the Entra ID portal, verify that the tenant ID is the same as in the `$tenantId` variable, because we will need that in the next set of PowerShell commands:

```
Disconnect-MgGraph
$credential = New-Object System.Management.Automation.PSCredential -ArgumentList
    $appRegistration.AppId, ($secret.SecretText | ConvertTo-SecureString -AsPlainText -Force)
Connect-MgGraph -TenantId $tenantId -ClientSecretCredential $credential
```

If you have followed and used the same PowerShell variables as I'm showing you, you will hopefully get a successful connection to Microsoft Graph.

The first command, `Disconnect-MgGraph`, will disconnect any ongoing sessions with Microsoft Graph; you can run the command twice just to verify that all sessions will be revoked.

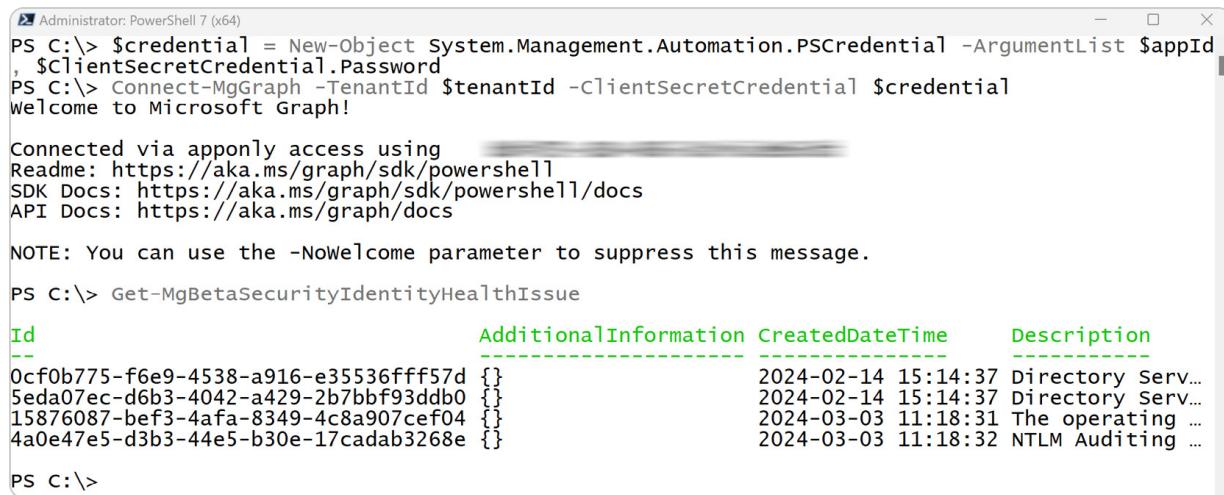
When `connect-MgGraph` has been successful, you will see the following output:

```
Welcome to Microsoft Graph!
Connected via app-only access using <GUID_of_appId>
Readme: https://aka.ms/graph/sdk/powershell
SDK Docs: https://aka.ms/graph/sdk/powershell/docs
API Docs: https://aka.ms/graph/docs
NOTE: You can use the -NoWelcome parameter to suppress this message.
```

Now, back to the main task; we wanted to see the health issues within the PowerShell session against Microsoft Graph, so let's start by typing in the following cmdlet:

```
Get-MgBetaSecurityIdentityHealthIssue
```

As you can see, in *Figure 3.10*, we have issues that we need to address.



ID	AdditionalInformation	CreatedDateTime	Description
0cf0b775-f6e9-4538-a916-e35536ffff57d	{}	2024-02-14 15:14:37	Directory Serv...
5eda07ec-d6b3-4042-a429-2b7bbf93ddb0	{}	2024-02-14 15:14:37	Directory Serv...
15876087-bef3-4afa-8349-4c8a907cef04	{}	2024-03-03 11:18:31	The operating ...
4a0e47e5-d3b3-44e5-b30e-17cadab3268e	{}	2024-03-03 11:18:32	NTLM Auditing ...

Figure 3.10 – List of health issues visualized in PowerShell

Now, you can drill into each of the issues with the following command:

```
(Get-MgBetaSecurityIdentityHealthIssue) [0] | Format-List
```

If you want to take a closer look, you will see a lot more properties, such as `HealthIssueType` , `Severity` , `Recommendations` , `SensorDnsNames` , and more.

We will do much more with this API later.

As we have started to dip our toes into API capabilities, we will now start looking at some of the automation use cases for MDI.

Automation in action – case studies and scripting scenarios

Considering the vast potential of scripting, cloud services, and architectural creativity, solutions can be both scalable and cost-effective. A crucial element of any MDI implementation is monitoring, essential for IT operations. While you might have different views on the approach I'm about to outline – and I respect differing opinions as each person and organization has its unique roles and responsibilities – I invite you to consider my perspective.

Responsibility for the sensor management, server configurations, and Group Policy settings should ideally fall within the IT operations team of the company. In smaller organizations, it's common for the same individual to manage server settings, configure cloud solutions, and even engage in threat hunting. As we explore various scenarios in this book, you'll have the opportunity to determine which team should receive alerts and who should handle the remediation of issues. This approach aims to streamline processes and enhance the operational efficiency of MDI systems across different organizational structures.

Monitoring the MDI service via Azure Monitor

Many organization servers are still in the on-premises world, but we can still use the power of innovations in the cloud. One product from Microsoft extends the on-premises world into the cloud, and multi-cloud, if you will, and that is Azure Arc.

TIERING AND AZURE ARC

The tiering model traditionally used for on-premise Active Directory infrastructures should be extended to your Azure environment to enhance security by clearly segmenting access levels and controlling privileged operations. By integrating Azure subscriptions, role-based access control (RBAC), and extension settings in the Azure Arc agent, you can better manage and secure your hybrid infrastructure. For example, by adding Tier 0 Azure Arc-enabled servers to a dedicated Azure subscription within the Platform Identity Management group (following Microsoft's enterprise-scale architecture), you automatically apply critical security measures, treating these assets with the same protections as any Tier 0 Azure virtual machines (VMs), safeguarding them from high-level threats.

With Azure Arc, we can install add-ons or extensions that really make the difference. One of those extensions is AMA. After we have onboarded the server through the Azure Arc agent, we can then, through different methods, install these extensions, such as AMA, or even Defender for Servers (`MDE.Windows` / `MDE.Linux`), and so on.

So, the scenario here is that we have onboarded our MDI sensor servers (domain controllers, AD FS, AD CS, and Entra Connect) with Azure Arc and have also deployed the AMA for monitoring. If you can create your own VM in a Hyper-V or VMware lab environment (or other hypervisor solution), please do – otherwise, we will use the lab in Azure with native VMs.

To install the solution for monitoring the MDI service, follow these steps:

1. Press the **Deploy to Azure** button (see *Figure 3.11*) for this scenario in the GitHub repository for this chapter (see the `Chapter03/1-MonitorMDIService` folder). You will need to sign in to Azure with an account that has appropriate permissions to create new resources.



Figure 3.11 – The Deploy to Azure button

2. Verify the deployment was successful and you can see all the resources in your resource group.

To start using the solution and to monitor the MDI service via the **Change Tracking and Inventory** solution, follow these steps:

1. Make sure that the **Change Tracking and Inventory** solution is installed on the Log Analytics workspace and that the **data collection rule (DCR)** is applied to the servers you listed in the array during the deployment.
2. Wait a few “Microsoft moments” (I know it’s hard to know how long, but be patient) so that every resource has the time to be fully provisioned in the backend.
3. Log in to one of the servers that has the MDI sensor installed, and then stop the MDI service (Azure Advanced Threat Protection Sensor) either manually or through PowerShell. Follow the coming steps to stop the service with PowerShell:
 - Start PowerShell as an administrator.
 - Type the following in the PowerShell window:

```
Stop-Service -Name AATPSensor
```

4. Go back to the Azure portal and to the Log Analytics workspace you deployed earlier.

5. Run the following KQL:

```
ConfigurationChange  
| where SvcName == "AATPSensor"
```

6. Adjust the query to your needs and then click on **+ New alert rule** within the Log Analytics pane.

7. Follow the steps on the page to create your own custom alert and be notified when the service stops.

Now that we can be alerted if the service suddenly stops so that we can take appropriate actions, it's time to see whether we can be alerted for MDI configuration drift.

Monitoring the MDI configuration with Azure Monitor and custom alert rules

As discussed in the previous section, we can make sure that our Group Policy and other local settings stay intact and if not, we receive alerts for any configuration drift. One way to monitor this is to look at the `.json` output from the MDI configuration report, the `New-MDIConfigurationReport` cmdlet. We monitor the specific folder and all the `.json` files with Azure Monitor DCRs and **data collection endpoints (DCEs)**, which make it possible for us to look at files in the operating system.

The logs will flow into Log Analytics, where we can trigger alerts as emails, a new case in our ITSM (IT service management) solution, or even an alert/incident in Microsoft Sentinel, which could also trigger some automation. If you have an on-premises environment, I highly recommend that you learn about Azure Arc, how to deploy it, and how to manage it. With Azure Arc, your on-premises servers will be just like any Azure resource in the Azure portal where you can install extensions (AMA, Defender for Servers, Azure Update Manager, etc.). But keep in mind that domain controllers, AD FS, AD CS, and Entra Connect servers are Tier 0 systems, and you need to close the gap so you don't get any lateral movement from the cloud to on-premises.

LEARN MORE ABOUT AZURE ARC SECURITY CONSIDERATIONS

Azure Arc security is a shared responsibility between Microsoft and the user, where Microsoft secures the cloud service and the user manages access and compliance. The Azure Connected Machine agent plays a central role in enabling Azure services on servers outside Azure data centers, with various built-in security controls to ensure robust protection. For more details, you can read the full article at Microsoft Learn: <https://learn.microsoft.com/en-us/azure/azure-arc/servers/security-overview>.

To install the solution for monitoring the MDI configuration, follow these steps:

1. Press the **Deploy to Azure** button (see *Figure 3.12*) for this solution in the GitHub repository for this chapter (see the `Chapter03/2-MonitorMDIConfiguration` folder). You will need to sign in to Azure with an account that has appropriate permissions to create new resources.



Figure 3.12 – The Deploy to Azure button

2. Verify the deployment was successful and you can see all the resources in your resource group.

To start using the solution and to monitor the MDI configuration via AMA, follow these steps:

1. Make sure that the AMA is fully installed and that you can see the new custom table in the specified Log Analytics workspace.
Also, verify that the DCR is applied to the servers you listed in the array during the deployment.
2. Wait a few *Microsoft moments* so that every resource has the time to be fully provisioned in the backend.
 - It is important that the DCR has at least one of the sensor servers in the **Resources** menu (see *Figure 3.13*).

Name	Type	Location	Data collection rule	Resource group
DC02	Virtual machine	West Europe	dce-westeu...	MDI

Figure 3.13 – Resources connected in a DCR

3. Log in to one of the servers with the MDI sensor installed and then run the PowerShell script in the **New-MDIConfigCheck.ps1** GitHub folder. This script will create a new function that can save the JSON output in one line. This is because of the current limitation in the AMA's handling of JSON files. For us, this means that we can append the file with new information, and it will be sent to Log Analytics. We will also create a log rotate function as an example. We will use the **Get-MDIConfiguration** cmdlet from the **DefenderForIdentity** module to get the current configuration:

```

function Format-JsonSingleLine {
    param(
        [Parameter(Mandatory, ValueFromPipeline)]
        [String] $json
    )
    # Remove all newline characters and unnecessary spaces
    $json = $json -replace '\s+', ' ' -replace '\s*([{}]\[.\]:,)\s*', '$1'
    return $json
}
function RotateLogFile {
    param(
        [Parameter(Mandatory)]
        [string]$filePath,
        [int]$maxFileSizeMB = 5
    )
    if (Test-Path $filePath) {
        $fileSizeMB = (Get-Item $filePath).Length / 1MB
        if ($fileSizeMB -ge $maxFileSizeMB) {
            Write-Host "File size is $fileSizeMB MB, exceeding $maxFileSizeMB MB."
            Deleting and recreating the file." -ForegroundColor Yellow
            Remove-Item $filePath -Force
        }
    }
}

```

```

        New-Item -ItemType File -Path $FilePath
    }
} else {
    Write-Host "File does not exist, creating a new one." -ForegroundColor Green
}
function Append-JsonLog {
    param(
        [Parameter(Mandatory)]
        [string]$FilePath,
        [Parameter(Mandatory)]
        [object]$Data,
        [int]$MaxFileSizeMB = 5
    )
    try {
        # Rotate log file if it exceeds the specified size
        Rotate-LogFile -FilePath $FilePath -MaxFileSizeMB $MaxFileSizeMB
        # Convert the data to JSON in a single line
        $json = $Data | ConvertTo-Json -Depth 5 | Format-JsonSingleLine
        # Append the JSON data to the file
        Add-Content -Path $FilePath -Value $json
        Write-Host "Appended JSON data to $FilePath" -ForegroundColor Green
    } catch {
        Write-Error "Failed to append JSON data: $_"
    }
}
# Import the necessary module
Import-Module DefenderForIdentity
# Fetch the MDI configuration data
$logData = Get-MDIConfiguration -Configuration All -Mode Domain
# Define the path for the log file
$logFilePath = "C:\Temp\MDIConfig\MDI-configuration.json"
# Append the log data to the file
Append-JsonLog -FilePath $logFilePath -Data $logData

```

4. Make a minor adjustment to the MDI configuration by modifying the **power performance** settings. You have two options:

- If you have a GPO for Power mode, do the following:
 - Navigate to **Computer Configuration > Policies > Administrative Templates > System > Power Management > Select an active power plan**.
 - Set the active power plan to **Power Saver**.
 - Close all dialogue boxes and then run **gpupdate /force** in an elevated PowerShell window.
 - Run the **powercfg /list** command in the same PowerShell window to verify that the mode changed.
- If you don't have a GPO for Power mode, follow these steps:
 - You can make this change directly using PowerShell. Run the following commands in an elevated PowerShell window:


```
powercfg /list
powercfg /setactive <GUID from the above command>
```

5. In the Azure portal, locate the Log Analytics workspace that you deployed and verify that you see some records in the custom table (for example, **MDIConfig_CL**) with KQL:

```
MDIConfig_CL
| take 1
```

The screenshot shows the Azure Log Analytics interface. The top navigation bar includes 'Log Analytics workspace' and tabs for 'Logs' and 'Metrics'. Below the navigation is a search bar with 'New Query 1*' and a 'Run' button. The main area has sections for 'Tables', 'Queries', 'Functions', and 'Custom Logs'. A 'Results' tab is selected, showing a table with three columns: 'TimeGenerated', 'RawData', and 'Type'. The 'RawData' column contains a JSON object: [{"Configuration": "CAAuditing", ...}]. The 'Type' column shows 'MDIConfig_CL'.

Figure 3.14 – KQL query Log Analytics

6. The result you see is that the entire JSON script is presented in the **RawData** column. We can parse that column with some KQL magic so that we can find the misconfiguration and be alerted when it happens. See the following KQL parser:

```
MDIConfig_CL
| take 1
| mv-expand ConfigItem = todynamic(RawData)
| project Configuration = ConfigItem.Configuration,
    Mode = ConfigItem.Mode,
    Status = ConfigItem.Status,
    DisplayName = tostring(ConfigItem.Details.DisplayName),
    Id = tostring(ConfigItem.Details.Id),
    GpoStatus = toint(ConfigItem.Details.GpoStatus),
    RegistryValue = ConfigItem.Details.RegistryValue.value,
    AuditSettings = ConfigItem.Details.AuditSettings.value,
    DetailsString = tostring(ConfigItem.Details)
| mv-expand RegistryItem = RegistryValue to typeof(dynamic)
| project Configuration, Mode, Status, DisplayName, Id, GpoStatus, AuditSettings,
    RegistryKeyName = tostring(RegistryItem.KeyName),
    RegistryValueName = tostring(RegistryItem.valueName),
    RegistryValue = tostring(RegistryItem.Value),
    RegistryValueDisplay = tostring(RegistryItem.valueDisplay),
    ExpectedRegistryValue = tostring(RegistryItem.ExpectedValue)
| mv-expand AuditItem = todynamic(AuditSettings)
| project Configuration, Mode, Status, DisplayName, Id, GpoStatus,
    PolicyTarget = tostring(AuditItem.PolicyTarget),
    SubcategoryName = tostring(AuditItem.SubcategoryName),
    SettingValue = tostring(AuditItem.SettingValue),
    ExpectedValue = tostring(AuditItem.ExpectedValue)
```

```

1 MDIConfig_CL
2 | take 1
3 | mv-expand ConfigItem = todynamic(RawData)
4 | project Configuration = ConfigItem.Configuration,
5 |     Mode = ConfigItem.Mode,
6 |     Status = ConfigItem.Status,
...

```

Configuration	Mode	Status ↑↓	DisplayName
> ProcessorPerformance	Domain	false	Microsoft Defender for Identity...
> CAAuditing	Domain	true	Microsoft Defender for Identity...
> NTLMAuditing	Domain	true	Microsoft Defender for Identity...
> NTLMAuditing	Domain	true	Microsoft Defender for Identity...

Figure 3.15 – KQL parser

Now, we can be alerted if the configuration required for the MDI sensor suddenly fails so that we can take appropriate actions. Let's look at another solution that we could use, and that is to send health issues and security alerts with syslog.

Sending health issues and security alerts via syslog to Microsoft Sentinel

We have an almost hidden setting within the Defender XDR portal under **System > Settings > Identities > Notifications > Syslog notifications**.

Here, we can enable the syslog service and when we do that, we will be asked to fill in which sensor server we will enable it on (as for now, we can only have one sensor connected to the syslog notification service), the endpoint for our syslog server together with a port, then choose which transport protocol (UDP/TCP), and lastly the format of the syslog (RFC 3164 or RFC 5424).

COMMON EVENT FORMAT (CEF)

The messages that the MDI syslog integration sends to SIEM (security information and event management) systems, such as Microsoft Sentinel, are formatted in CEF.

After configuring the syslog service, select the types of notifications to send to your syslog server, including whenever the following occurs:

- A new security alert is detected.

- An existing security alert is updated.
- A new health issue is detected.

To install the solution for monitoring health issues and security alerts with syslog servers, follow these steps:

1. Press the **Deploy to Azure** button (see *Figure 3.16*) for this solution in the GitHub repository for this chapter (see the **Chapter03/3-HealthIssuesSyslog** folder). You will need to sign in to Azure with an account that has appropriate permissions to create new resources.



Figure 3.16 – The Deploy to Azure button

2. Verify the deployment was successful and you can see all the resources in your resource group.

To start using the solution and to monitor MDI health issues and security alerts with syslog, follow these steps:

1. Start by verifying that the syslog server is installed correctly and that you can access it. Also, verify that the **AzureMonitorLinuxAgent** and **syslog-ConfigScript** extensions don't have any errors (you will see that under the **Extensions + Applications** submenu of the VM in Azure).
2. Log in to the Linux VM using the username and password you set during deployment. You can do this from your terminal or PowerShell by running the following command:

```
ssh username@ip_address
```

Replace **username** with the actual username and **ip_address** with the IP address of the Linux VM.

3. After running the previous command, type in your password.
4. Now, let's verify that the Azure Monitor service is up and running by typing the following command in the PowerShell window, which is connected to the Linux server (see *Figure 3.17*):

```
systemctl status azuremonitoragent
```

```

azureuser@vm-syslog-0:~$ systemctl status azuremonitoragent
● azuremonitoragent.service - Azure Monitor Agent daemon (on systemd)
  Loaded: loaded (/etc/systemd/system/azuremonitoragent.service; enabled; vendor
  Active: active (running) since Fri 2024-08-23 09:34:17 UTC; 1 day 8h ago
    Docs: man:azuremonitoragent(8)
   Main PID: 873 (mdsd)
      Tasks: 86 (limit: 9506)
     Memory: 559.3M (max: 10.0G available: 9.4G)
       CPU: 22min 10.988s
      CGroup: /system.slice/azuremonitoragent.service
              └─873 /opt/microsoft/azuremonitoragent/bin/mdsd -A -c /etc/opt/microsoft/azur
Aug 23 09:34:13 vm-syslog-0 systemd[1]: Starting Azure Monitor Agent daemon (on syste>
Aug 23 09:34:13 vm-syslog-0 azuremonitoragent[593]: * Starting Azure Monitor Agent >
Aug 23 09:34:17 vm-syslog-0 azuremonitoragent[593]: ..done.
Aug 23 09:34:17 vm-syslog-0 systemd[1]: Started Azure Monitor Agent daemon (on syste>
[Lines 1-15/15 (END)]

```

Figure 3.17 – Verify the azuremonitoragent service

5. Go back to the Azure portal and the Log Analytics workspace that you referenced during the deployment, then run the following KQL query to verify that messages are sent from the Linux VM to Log Analytics:

```

Syslog
| where Computer == "ServerName"
| summarize by HostName

```

6. If you haven't already, it's time to configure MDI to send syslog messages. Log in to the Defender XDR portal and navigate to **System > Settings > Identity > Notifications > Syslog notifications**, or go to the direct link:
<https://security.microsoft.com/securitysettings/identities?&tabid=siem> :

- Enable the syslog service
- Press **Edit configurations**
 - **Sensor** : Choose one of the sensors
 - **Service endpoint** : Type the private IP address of the Linux VM
 - **Port** : 514
 - **Transport** : TCP
 - **Format** : RFC 5424

7. Click **Save**.

8. Enable the notifications that you want to send with syslog:

- A new alert is detected.
- An existing security alert is updated.
- A new health issue is detected.

9. Click **Save**.

DIFFERENCE BETWEEN RFC 3164 AND RFC 5424

RFC 3164 defines the original syslog protocol, which is simpler and less structured. It uses a basic format for log messages, focusing on backward compatibility. However, it lacks some modern features, such as structured data, and is limited in terms of message length and time precision. Here's how you can send a test message:

```
logger "Test message in RFC 3164 format"
```

RFC 5424 modernizes syslog, offering a more flexible and standardized format. It introduces structured data elements, better timestamping with higher precision, and improved internationalization support. Here's how you can send a test message:

```
logger --rfc5424 "Test message in RFC 5424 format"
```

Now, we have configured three different automation scenarios where we can take the power of the cloud to gather more information and get help with the operation of our MDI sensors. I highly recommend you start thinking about your own type of solutions, what type of products or services can you combine, or even build more on top of the three solutions you just read about.

Let's move forward to the *Summary* section to see what we accomplished in this chapter.

Summary

In this chapter, we took the first step into the MDI PowerShell module. We explored how to install this module on servers, even those with restricted internet access, and delved into the capabilities of the module.

Next, we turned our attention to the health issues API within the Microsoft Graph API. We learned how to retrieve health issues from our MDI sensor servers, equipping us with the necessary information for effective remediation.

Finally, you were presented with several automation scenarios to explore. I encourage you to investigate these further and consider their potential implementation within your organization.

In the next chapter, we will learn how to integrate the MDI sensors with AD FS, AD CS and Entra Connect.

Part 2: Advanced Configuration, Integration, and Threat Detection

In this part, we explore how to leverage **Microsoft Defender for Identity (MDI)** to its full potential by diving into advanced configurations and integrations. You'll learn to connect MDI with essential services, such as AD FS, AD CS, and Entra Connect, for comprehensive identity protection. Additionally, we will examine how to extend MDI capabilities using APIs and apply advanced KQL techniques for in-depth threat detection.

This part includes the following chapters:

- [Chapter 4](#), *Integrating MDI with AD FS, AD CS, and Entra Connect*
- [Chapter 5](#), *Extending MDI Capabilities Through APIs*
- [Chapter 6](#), *Mastering KQL for Advanced Threat Detection in MDI*

4

Integrating MDI with AD FS, AD CS, and Entra Connect

In this chapter, we will delve into the integration of **Microsoft Defender for Identity (MDI)** with key Active Directory services – specifically, **Active Directory Federation Services (AD FS)**, **Active Directory Certificate Services (AD CS)**, and **Entra Connect**. Integrating MDI with services such as AD FS, alongside data from our domain controllers, enhances our capability to correlate login data extensively. This allows for a deeper analysis of user behavior and authentication patterns, providing a more detailed and enriched security overview.

If you are thinking “*Why the support of AD FS; shouldn’t we migrate our apps to Entra ID?*”, the answer is yes, we should, but in some scenarios and some cases, we still need to have our AD FS infrastructure alive.

Additionally, we explore how to extend MDI’s coverage across multiple Active Directory forests, a step that is vital for organizations managing complex network architectures. This extension ensures consistent security policies and practices are applied universally, no matter the geographical or logical distribution of resources.

The chapter also covers the integration of MDI with **Virtual Private Networks (VPNs)**, highlighting strategies to secure remote activities and protect data flows. This is increasingly important today because of the mobile-first, work-from-anywhere, hybrid cloud environment era, and just in the last couple of years, there have been a lot of vulnerabilities in VPN solutions.

By the end of this chapter, you will understand how to effectively integrate MDI with these essential services and configure settings to optimize your security stance. You will be equipped to enhance identity protection within your organization, secure and manage cross-service communications and data transfers, and apply consistent security measures across distributed IT environments.

In this chapter, we’re going to cover the following main topics:

- Integrating MDI with AD FS
- Integrating MDI with AD CS
- Integrating MDI with Entra Connect
- Expanding MDI across multiple Active Directory forests
- VPN integration – securing remote activities and data flows

Let’s get started!

Technical requirements

Completion of [Chapter 1](#) and [Chapter 2](#) is a prerequisite for this section, and as a reminder, you also require the following:

- Microsoft tenant
- Microsoft subscription that includes Microsoft Defender for Identity, such as Microsoft 365 E5 or Microsoft 365 E3 + E5 Security
- Basic Microsoft 365 knowledge
- Active Directory knowledge
- A virtual or physical server environment with Active Directory installed and configured:
 - Optional: Active Directory Federation Services and Active Directory Certificate Services installed and configured
- Basic PowerShell knowledge:
 - Windows PowerShell 5.1 or PowerShell 7.4 or later
- Basic networking knowledge
- VPN solution:
 - Optional: Have a **Routing and Remote Access Server (RRAS)** ready to be configured and integrated into MDI

All the code examples for this chapter can be found on GitHub at

<https://github.com/PacktPublishing/Microsoft-Defender-for-Identity-in-Depth/tree/main/Chapter04>

Integrating MDI with AD FS

Before we go in on how the MDI sensor will be installed and what needs to be configured on an AD FS server, I want you to understand the basics of how AD FS works.

AD FS enables federated identity and access management by allowing authentication across different organizational boundaries. It's designed to facilitate **Single Sign-On (SSO)**, allowing users to access multiple applications on different networks using a single set of credentials, thereby enhancing both user convenience and security.

At the beginning of the Microsoft Cloud era with Office 365 (now Microsoft 365), it was very common that hybrid identity implementations were using AD FS together with Web Application Proxy and DirSync/Azure AD Connect. This setup was called **Federated Identity**.

Let's not stay in the old days; many of the services have been renamed and rebranded, so I do hope you are following. While **Password Hash Synchronization (PHS)** and **Pass-Through Authentication (PTA)** in Azure AD Connect replaced the need for AD FS for most organizations, AD FS continued to be used, particularly where **Certificate-Based Authentication (CBA)** was required. Now that CBA is supported in Entra ID, many organizations can finally transition away from AD FS.

DirSync became *Azure AD Connect*, and then *Microsoft Entra Connect*.

Office 365 became *Microsoft 365*.

A common standard deployment topology of AD FS will include the following:

- One or more AD FS servers in the internal corporate network.
- One or more **Web Application Proxy (WAP)** servers positioned in a DMZ (demilitarized zone) or extranet network, or a third-party reverse proxy solution such as F5 BIG-IP or Azure Application Gateway.

INFO

If you look at the GitHub repo for the first chapter, you will find a Bicep file for deploying the standard topology of AD FS and WAP in Azure. Please note that this is only for lab purposes.

As you may understand, AD FS, like our domain controllers, is considered and classified as a Tier 0 asset in the context of security and operational importance. The compromise of either a domain controller or an AD FS server can have severe consequences, potentially allowing unauthorized access to a wide array of critical resources and data. Thus, both require the highest levels of security measures, monitoring, and management to ensure the safety and stability of an organization's IT environment.

IMPORTANT NOTE

Make sure to follow hardening best practices for your AD FS deployment; see Microsoft Learn for more information:
<https://learn.microsoft.com/en-us/windows-server/identity/ad-fs/deployment/best-practices-securig-ad-fs>.

How AD FS authentication works

By including WAP in the authentication process, we add an additional layer of security by ensuring that the AD FS server is not exposed directly to the Internet. WAP handles all incoming requests and pre-authenticates user requests before they reach the internal AD FS server. Let's look at the flow:

1. **Authentication request**: When a user attempts to access a federated application, instead of directly entering credentials into the application, they are redirected to the WAP server.
2. **Pre-authentication at WAP**: The WAP server, located in the organization's DMZ, acts as the first checkpoint. It receives the authentication request and performs initial pre-authentication, ensuring that only secure and valid requests are forwarded to the AD FS server.
3. **Token generation at AD FS**: Once the request is considered secure, WAP forwards it to the AD FS server. Here, the user is prompted to enter their credentials. The AD FS server verifies these credentials against Active Directory.
4. **Token transmission to WAP**: The security token is relayed back to the WAP server, which then forwards it to the user's browser.
5. **Token consumption by browser**: The user's browser uses this token to access the requested federated application. The application inspects the token, and if the claims are validated, it grants access to the user.

6. Authorization and access control : Depending on the claims in the token and the application's policy, the user may have different levels of access or capabilities within the application.

See *Figure 4.1* for an illustration of the AD FS authentication process.

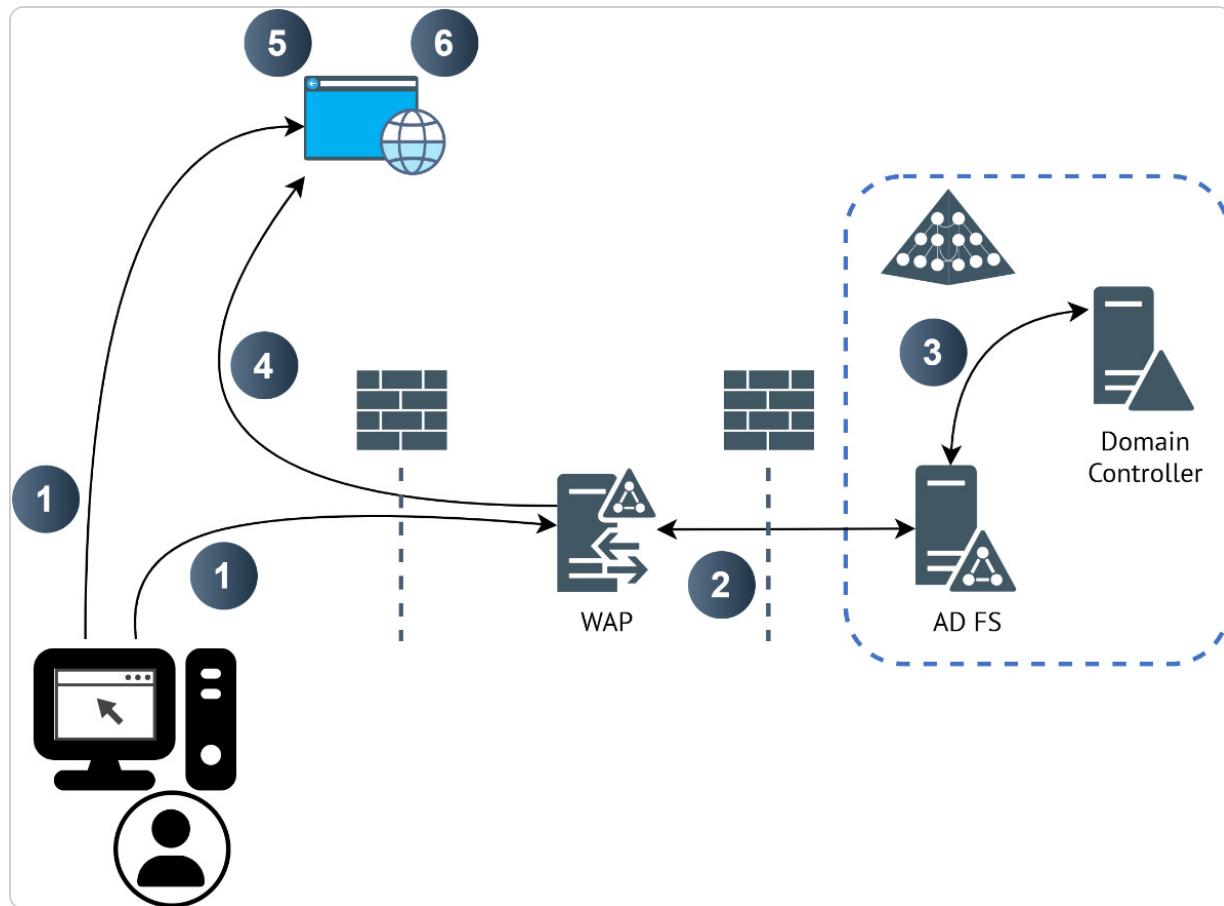


Figure 4.1 – AD FS authentication process

Let's move on to configuring the MDI sensor in our AD FS server.

Configuring AD FS for MDI sensor installation

Before we install the MDI sensor on our AD FS servers, I think we need to check our current configuration with the PowerShell module.

If you haven't installed or imported the PowerShell module on the servers, please do so and follow the guide in [Chapter 2](#).

We have two configurations that need to be done on each AD FS server:

- AD FS auditing
- Configure the MDI directory services account to read the AD FS database

IMPORTANT NOTE

In AD FS environments, the MDI sensor is only supported on federation servers and does not need to be installed on WAP servers.

AD FS advanced auditing/verbose settings

The following AD FS events are required for the MDI installation and configuration:

- **1202:** The Federation Service validated a new credential
- **1203:** The Federation Service failed to validate a new credential
- **4624:** An account was successfully logged on
- **4625:** An account failed to log on

With the MDI PowerShell module, we can quickly configure extended audit settings. Follow these steps to configure the audit settings for the event logs:

1. On your AD FS server, do the following:

- Start Windows PowerShell.
- Ensure that the **DefenderForIdentity** module is installed on your server. For detailed installation instructions, see [Chapter 2](#).
- On servers that are not domain controllers, ensure the following modules are installed, as they are required by the **DefenderForIdentity** module:
 - **ActiveDirectory** (**RSAT-AD-Tools**):

```
Install-WindowsFeature -Name RSAT-AD-Tools
• GroupPolicy (GPMC):
```

```
Install-WindowsFeature -Name GPMC
```

2. Type the following commands to configure the required audit settings:

```
Import-Module -Name DefenderForIdentity
Set-AdfsProperties -AuditLevel Verbose
```

Next, we need to configure the required **system access control list (SACL)** on the AD FS container in Active Directory.

Configuring the SACL on the AD FS container

We have two different ways to handle the SACL configuration, either via PowerShell and the MDI module or via the Active Directory Users and Computers snap-in. Let's look at these next.

PowerShell

The following is a snippet from the MDI PowerShell module, outlining the necessary configurations. Let's examine each attribute and its corresponding value:

```
$AdfsAuditing = @{
    Validate = 'LDAP://CN=ADFS,CN=Microsoft,CN=Program Data,{0}'
    Path     = 'AD:\CN=ADFS,CN=Microsoft,CN=Program Data,{0}'
    Auditing = '@'
    SecurityIdentifier,AccessMask,AuditFlagsValue,AceFlagsValue,InheritedObjectAceType,InheritanceType,PropagationFlags
    S-1-1-0,48,3,194,00000000-0000-0000-0000-000000000000,1,0
    '@ | ConvertFrom-Csv
}
```

The `validate` and `path` objects hold an LDAP (Lightweight Directory Access Protocol) and an Active Directory path for validation purposes. Those objects are within the bigger `settings` object, which will then be used to gather specific information, as follows:

```
Get-MDIAdPath -Path $settings.AdfsAuditing.Path
```

In the following code, we have the `Get-MDIAdPath` helper function that collects and tries to find the path for each of the configurations we can set with the MDI PowerShell module:

```
function Get-MDIAdPath {
    param(
        [Parameter(Mandatory)] $Path
    )
    $DefaultNamingContext = ([adsi]('LDAP://{0}/RootDSE' -f
$env:USERDNSDOMAIN)).defaultNamingContext.Value
    $Path -f $DefaultNamingContext
}
```

We have the following helper functions (that use other functions as well) to collect, verify, and set the appropriate auditing settings, respectively:

- `Get-MDIAdfsAuditing`
- `Test-MDIAdfsAuditing`
- `Set-MDIAdfsAuditing`

TIP

Look at `DefenderForIdentity.psm1`, located in the PowerShell module path (see [Chapter 3](#), in the section called *Installing MDI PowerShell module for the folder path*) to learn more about the helper functions used in the MDI PowerShell module.

In the following list, you will learn about the key attributes contained within the `auditing` object. Each attribute plays a specific role in defining how audit policies are applied, specifying which user groups are included, the permissions monitored, and how these settings propagate within the system:

- `SecurityIdentifier (SID) - S-1-1-0 :`
 - This SID represents the `Everyone` group, which includes all users.

- **AccessMask** – 48 :

- This is a bitmask that specifies the permissions that are audited. A value of 48 typically indicates permissions such as read (bitmask value of 32) and execute (bitmask value of 16). In this scenario, it is for the **Read all properties** and **Write all properties** permissions.

- **AuditFlagsValue** – 3 :

- This integer represents the type of auditing to be applied. A value of 3 means both success and failure events are to be audited.

- **AceFlagsValue** – 194 :

- These are the flags that provide additional information about the condition under which the **access control entry (ACE)** that controls auditing and inheritance applies. 194 indicates *All access attempts are audited* and *The access mask is propagated to child container objects*.

- **InheritedObjectType** – 00000000-0000-0000-0000-000000000000 :

- This **Globally Unique Identifier (GUID)** is all zeros, which means that there isn't a specific object type that this audit policy entry is targeting – it applies generally rather than being restricted to a particular type of object.

- **InheritanceType** – 1 :

- This indicates how the audit setting is inherited. 1 signifies that the settings apply to *all* and, in this case, directly to the object and the object's immediate children, and the descendants of the object's children, and in the UI, it will be the **This object and all descendant objects** setting.

- **PropagationFlags** – 0 :

- This refers to how permissions are passed down to subfolders or files (child objects) in a directory or system. When the value is set to 0, it means there are no special rules for passing down these permissions. In simple terms, the permissions that apply to the current folder or file won't be automatically applied to its subfolders or files unless manually specified.

Follow the next steps to configure the SACL for advanced auditing settings:

1. On one of your domain controllers, start Windows PowerShell.
2. Run the following command for configuring the AD FS auditing settings:

```
Set-MD IConfiguration -Mode Domain -Configuration AdfsAuditing
```

3. Verify the configuration with the following command; you should now get an output that says **True** :

```
Test-MD IConfiguration -Mode Domain -Configuration AdfsAuditing
```

If you are more comfortable with doing the configuration in the user interface instead, you can follow the next steps.

Active Directory Users and Computers

Follow the next steps to configure the SACL for advanced auditing settings via the UI:

1. On one of your domain controllers, start the Active Directory Users and Computers snap-in.
2. Click on **View** and then on **Advanced Features**.
3. Expand the domain that you want to configure – in my case, I'm configuring the `contoso.local` domain.
4. Navigate to **Program Data > Microsoft > ADFS**, right-click on the **ADFS** container, and choose **Properties** (see *Figure 4.2*).



Active Directory Users and Computers

File Action View Help



- Active Directory Users and Computers
- > Saved Queries
- > contoso.local
 - > Builtin
 - > Computers
 - > Domain Controllers
 - > ForeignSecurityPrincipals
 - > Keys
 - > LostAndFound
 - > Managed Service Accounts
 - > MDETest
 - > Program Data
 - > Microsoft
 - > ADFS
 - > Servers
 - > Service Accounts
 - > System
 - > Users
 - > NTDS Quotas
 - > TPM Devices

Name	Type
CryptoPolicy	Contact
ea469d7e-8b...	Container

Figure 4.2 – ADFS container in Active Directory

5. Navigate to the **Security** tab and then select **Advanced > Advanced Security Settings > Auditing tab > Add > Select a principal** .

6. Under **Enter the object name to select** , type **Everyone** .

7. Click on **Check Names** and then on **OK** .

8. You then return to **Auditing Entry for ADFS** . Make sure to select the following:

- **Type : All**
- **Applies to : This object and all descendant objects**
- **Permissions** : Scroll down to click on **Clear all** , then scroll up and select **Read all properties** and **Write all properties** (see *Figure 4.3*).

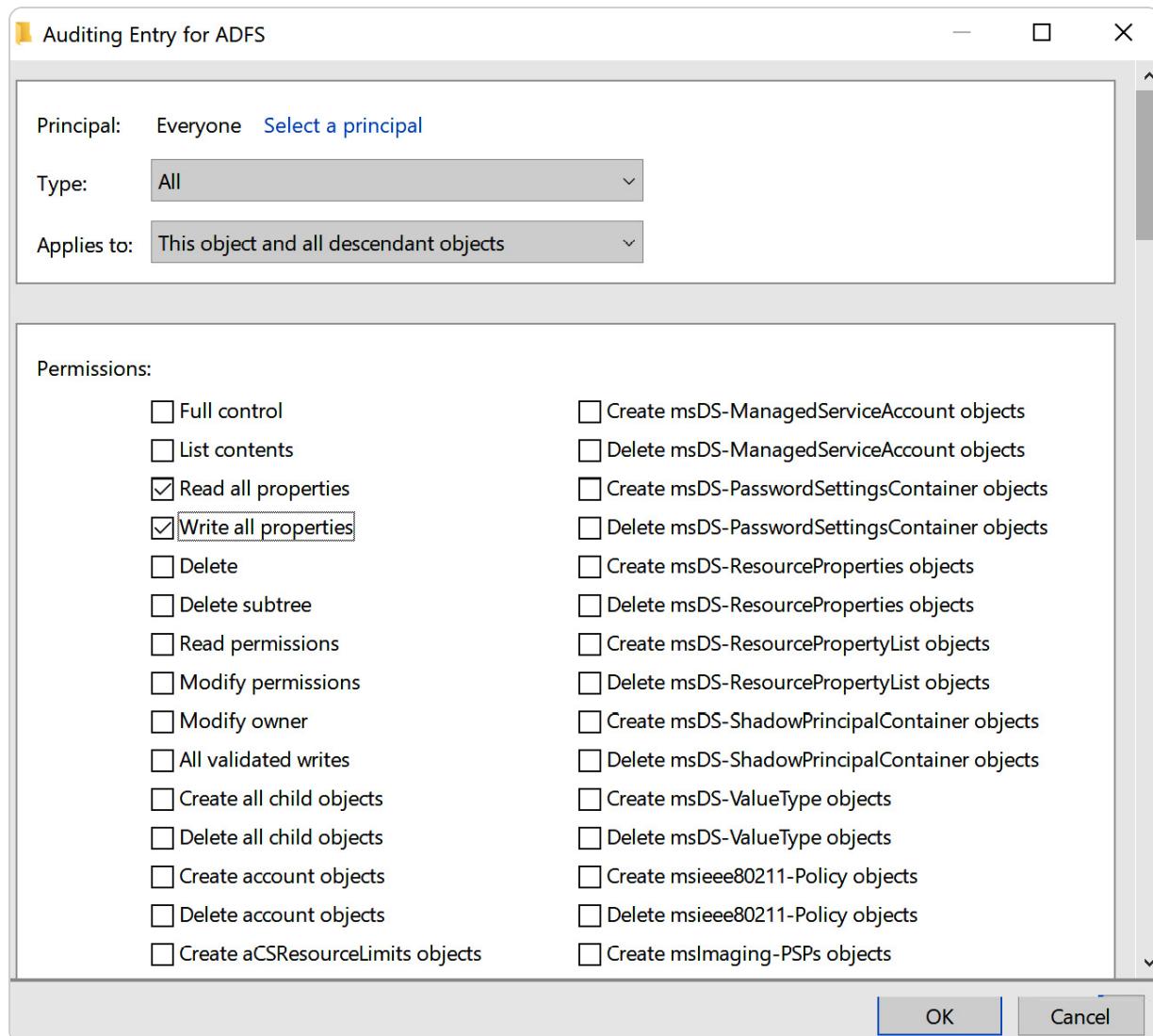


Figure 4.3 – Auditing entry for ADFS

9. Now, click on **OK** to save the configuration

10. You can use the MDI PowerShell module to verify the setting that we just configured. Do so by starting Windows PowerShell, and if you haven't already, please install the necessary modules, `ActiveDirectory` and `GroupPolicy`, and run the following command from the MDI PowerShell module. You should get an output that says `True`:

```
Install-WindowsFeature -Name RSAT-AD-Tools  
Install-WindowsFeature -Name GPMC  
Test-MDIClusterConfiguration -Mode LocalMachine -Configuration AdfsAuditing
```

Now that we have configured the correct SACL for the advanced auditing settings, we are ready to configure the AD FS database permissions.

AD FS database permissions

When we install AD FS from scratch, we need to make the decision of either using the **Windows Internal Database (WID)** or Microsoft SQL Server. The following section may differ a bit based on the choice of database type. You may ask your **database administrator (DBA)** to help if you are uncomfortable with managing the upcoming settings when it comes to Microsoft SQL Server.

INFO

For clarity and simplicity, this book will focus exclusively on configurations using the WID instead of Microsoft SQL Server due to its complexity. Learn more about ADFS and SQL Server at the following link: <https://learn.microsoft.com/en-us/windows-server/identity/ad-fs/technical-reference/the-role-of-the-ad-fs-configuration-database>.

In short, our Directory Service account needs the following permissions to the AD FS `configuration` database:

- `log in`
- `read`
- `connect`
- `select`

The AD FS `configuration` database contains critical configuration information about the Federation Service, including trust relationships, policy settings, and more.

MDI utilizes this access to extract and understand configuration details that impact how authentication requests are handled within the Federation Service. This data is crucial for MDI to accurately assess potential security threats, detect anomalies, and provide insights into unauthorized or malicious activities. By having read permissions, the Directory Service account allows MDI to securely gather the necessary information without modifying any data, thereby maintaining the integrity of the AD FS environment while still enabling comprehensive security monitoring.

TIP

If we are using Microsoft SQL Server, we can obtain the SQL connection string by some simple and short PowerShell commands.

Open Windows PowerShell on your federation server and type the following:

```
$adfs = gwmi -Namespace root/ADFS - Class SecurityTokenService
```

You have now obtained the connection string to the AD FS database, and to view the value, you simply enter the following:

```
$ adfs.ConfigurationConnectionString
```

If we are using WID, we need to first gather the database name, insert that name as a replacement of the `OurDatabaseName` value used in this example, and then we can use the following line to form our connection string, saved in the variable called `$connectionString`. Here's some PowerShell magic to find the database name:

```
$adfs = Get-WmiObject -Namespace "root/ADFS" -Class "SecurityTokenService"
$connectionString = $adfs.ConfigurationConnectionString
$components = $connectionString.Split(',')
$initialCatalogComponent = $components | Where-Object { $_.StartsWith("Initial Catalog") }
$initialCatalog = $initialCatalogComponent.Split('=')[1].Trim()
Write-Output "Connection String: $connectionString"
Write-Output "Initial Catalog: $initialCatalog"
```

Now, we can continue with setting the permissions on the AD FS database. Don't forget to correctly type in the gMSA account if you created that earlier; the correct format will then be

```
DOMAIN\GMSA_ACCOUNT$ :
```

```
$ConnectionString =
'server=\\.\\pipe\\MICROSOFT##WID\\tsql\\query;database=OurDatabaseName;trusted_connection
=true;'
$SQLConnection= New-Object System.Data.SqlClient.SqlConnection($ConnectionString)
$SQLConnection.Open()
$SQLCommand = $SQLConnection.CreateCommand()
$SQLCommand.CommandText = @@
USE [master];
CREATE LOGIN [CONTOSO\MDIGMSA$] FROM WINDOWS WITH DEFAULT_DATABASE=[master],
USE [OurDatabaseName];
CREATE USER [CONTOSO\MDIGMSA$] FOR LOGIN [CONTOSO\MDIGMSA$];
ALTER ROLE [db_datareader] ADD MEMBER [CONTOSO\MDIGMSA$];
GRANT CONNECT TO [CONTOSO\MDIGMSA$];
GRANT SELECT TO [CONTOSO\MDIGMSA$];
"@
$SqlDataReader = $SQLCommand.ExecuteReader()
$SQLConnection.Close()
```

Now that we have given the MDI Directory Service Account (DSA) (the gMSA account) the right permissions, we should verify that the sensor is sending events to the MDI service.

Validating the AD FS integration

To be able to verify that the events are sent to the MDI service, we need to do the following:

1. Verify that the sensor service is started and running using Windows PowerShell (or PowerShell 7 if you have installed that on the federation server):

```
Get-Service AATPSensor | Select-Object Status
```

The output of the preceding command should be `Running`.

2. Make sure that the `IdPInitiatedSignonPage` property is enabled on your AD FS farm, which will help us test the sign-ins. If you get `False` as a result of the first command, set the property with the `Set-AdfsProperties` cmdlet as shown here:

```
Get-AdfsProperties | Select-Object -ExpandProperty  
EnableIdPInitiatedSignonPage  
Set-AdfsProperties -EnableIdPInitiatedSignonPage  
$true
```

3. Go to your AD FS URL (you can see the URL/hostname in the `Get-AdfsProperties` output) such as `https://adfs.contoso.local/adfs/ls/idpinitiatedsignon`, and sign in with your admin account. If we have configured everything correctly, the MDI sensor will be able to read from the AD FS database.
4. If not logged in already, please log in to security.microsoft.com, and from the menu blade, choose **Investigation & response > Hunting > Advanced hunting**.
5. Enter the following query in the **New query** tab:

```
IdentityLogonEvents | where Protocol =~ 'Adfs'
```

You should now see some rows with information, and if not, you must check and verify the settings (see *Figure 4.4*).

The screenshot shows the Microsoft Advanced hunting interface. At the top, there's a navigation bar with 'New query*', a '+' button, 'Run query' (which is currently selected), 'Last 3 hours', 'Save', 'Share link', and 'Manage rules'. Below the navigation is a 'Query' section containing the following PowerShell-like query:

```
1 IdentityLogonEvents  
2 | where Protocol =~ "Adfs"  
3 | project TimeGenerated, LogonType, Protocol, ActionType, Application
```

Below the query, there are three tabs: 'Getting started' (disabled), 'Results' (selected), and 'Query history'. The 'Results' tab shows a table with two items. The columns are: TimeGenerated, LogonType, Protocol, ActionType, and Application. The data in the table is as follows:

TimeGenerated	LogonType	Protocol	ActionType	Application
> Aug 27, 2024 10:0...	Logon with ADFS authentication	Adfs	LogonSuccess	Active Dire...
> Aug 27, 2024 10:0...	Logon with ADFS authentication	Adfs	LogonSuccess	Active Dire...

Figure 4.4 – Verifying data ingestion in Advanced hunting

Now, we are ready to tackle the integration of MDI on AD CS.

Integrating MDI with AD CS

Just as you learned some fundamental knowledge about AD FS, we will start off this section by taking a few steps back to give you a high-level overview of AD CS.

Let's start by looking at the most common terms when it comes to AD CS:

- **Active Directory Certificate Services (AD CS)**: Microsoft's PKI implementation.
- **Certificate Authority (CA)**: PKI server that issues certificates.
- **Certificate template** : A collection of settings and policies that define the contents of a certificate issued by an enterprise CA.
- **Certificate Signing Request (CSR)**: A message sent to a CA to request a signed certificate.
- **Extended/Enhanced Key Usage (EKU)**: One or more **object identifiers (OIDs)** that define how a certificate can be used.
- **Enterprise CA** : CA integrated with AD (as opposed to a standalone CA); offers certificate templates
- **Public Key Infrastructure (PKI)**: A system to manage certificates/public key encryption

For an enterprise using Active Directory and planning to implement AD CS, setting up a tiered CA structure is recommended to enhance security. This setup involves at least two levels of CAs: a top-level or Root CA and one or more Subordinate CAs. The Root CA is kept secure and doesn't directly interact with computers on the network, greatly reducing the risk of security breaches. The Subordinate CAs handle the day-to-day issuance and management of certificates, making it easier to control and secure certificate issuance across different parts of the organization. To effectively implement a tiered CA structure and ensure robust security, several key components and practices must be established:

- **Offline Root CA** : Set up an offline Root CA to issue certificates only to one or more subordinate CAs. This CA should be securely stored and rarely brought online, only as needed for specific tasks such as issuing new subordinate CA certificates or renewing existing ones.
- **Subordinate Enterprise CAs** : Deploy one or more Enterprise CAs that are integrated with Active Directory. These CAs handle the day-to-day issuance, renewal, and revocation of certificates for users, computers, and services within the organization.
- **Configure certificate templates** : Define and configure certificate templates according to organizational needs and security policies. Ensure templates are set up with the appropriate permissions and usage properties to prevent unauthorized issuance.
- **Implement strong authentication and authorization policies** : Enforce strict authentication and authorization policies for certificate issuance to minimize the risk of unauthorized certificate access or misuse.

This structure not only secures the CA hierarchy but also leverages Active Directory for authentication, making certificate management efficient and secure, if we configure everything right.

RECOMMENDED READING

I encourage you to read the blog post and whitepaper called *Certified Pre-Owned: Abusing Active Directory Certificate Services*, by Will Schroeder and Lee Christensen. They are really raising the awareness of how AD CS can be abused because of this complex product. Here is the link to the blog: <https://blog.qdsecurity.se/2020/09/04/supply-in-the-request-shenanigans/>.

This is the link to the whitepaper: https://specterops.io/wp-content/uploads/sites/3/2022/06/Certified_Pre-Owned.pdf.

As you can envision, AD CS is also a Tier 0 asset and requires robust protection like domain controllers or federation servers. It's crucial to follow strict guidelines on how certificate templates should be structured and configured. For instance, you should avoid settings that allow the **Subject Alternative Name (SAN)** to be specified during the request (with the setting called **Supply in the request**) because this could enable users to misuse the **User Principal Name (UPN)** field. A misconfigured template could let an end user request a certificate with a domain admin account in the SAN, leading the CA to inadvertently sign a potentially malicious certificate.

In the lab environment, I have added a few vulnerable templates and settings in AD CS that MDI will find and present under the **Microsoft Secure Score** section in the Defender XDR portal (see *Figure 4.5*).

Microsoft Secure Score

Overview **Recommended actions** History Metrics & trends

Actions you can take to improve your Microsoft Secure Score. Score updates may take up to 24 hours.

 Export

Filters: Product: Defender for Identity 

Rank  Recommended action 

- | | | |
|--------------------------|----|---|
| <input type="checkbox"/> | 1 | Resolve unsecure domain configurations |
| <input type="checkbox"/> | 2 | Disable Print spooler service on domain controllers |
| <input type="checkbox"/> | 3 | Protect and manage local admin passwords with Microsoft LAPS |
| <input type="checkbox"/> | 4 | Edit misconfigured certificate templates ACL (ESC4) |
| <input type="checkbox"/> | 5 | Edit misconfigured enrollment agent certificate template (ESC3) |
| <input type="checkbox"/> | 6 | Edit overly permissive Certificate Template with privileged EKU (Any purpose EKU or No EKU) (ESC2) |
| <input type="checkbox"/> | 7 | Prevent users to request a certificate valid for arbitrary users based on the certificate template (ESC1) |
| <input type="checkbox"/> | 8 | Edit misconfigured Certificate Authority ACL (ESC7) |
| <input type="checkbox"/> | 9 | Edit vulnerable Certificate Authority setting (ESC6) |
| <input type="checkbox"/> | 10 | Edit insecure certificate enrollment IIS endpoints (ESC8) |

Figure 4.5 – Secure Score

Before we dive into how MDI and AD CS can work together, let's learn about the AD CS key processes.

How AD CS works

The following is an outline of the key processes involved in the lifecycle of certificates managed by AD CS:

1. **Role installation and configuration** : Initially, AD CS is installed as a role on a server within the network. The configuration involves setting up a CA, which will be responsible for issuing and managing certificates. The CA can be configured as a Root CA or a subordinate CA depending on the hierarchy required.
2. **Certificate templates configuration** : Administrators configure certificate templates that define the settings and properties of the certificates that the CA will issue. These templates include information such as the certificate's validity period, usage purposes (e.g., encryption, digital signatures), and renewal policies. MDI will help us by looking at these templates to see whether they are subject to an escalation attack.
3. **Certificate enrollment process** : Users or devices request certificates from the CA through various enrollment methods, such as a web enrollment interface, auto-enrollment via Group Policy for domain-joined machines, or manual enrollment by submitting a CSR to the CA.
4. **Certificate issuance** : Upon receiving a request, the CA verifies the identity and eligibility of the requester based on predefined policies. If the request meets all the criteria, the CA issues a certificate, attaching the requester's public key to it and signing it with the CA's private key to validate its authenticity.
5. **Certificate retrieval** : Once issued, the certificate is made available to the requester. It can be installed automatically on the user's device or downloaded manually through a web portal, depending on the enrollment method used.
6. **Usage of certificates** : Certificates are used for various purposes such as securing email communications, authenticating to services, encrypting files or network traffic, and signing code or documents to prove integrity and origin.
7. **Certificate renewal and revocation** : Certificates have a finite validity period and need to be renewed periodically. AD CS automates the renewal process for certificates enrolled through auto-enrollment. Additionally, if a certificate is compromised or no longer needed, it can be revoked, and the revocation status is updated in the **Certificate Revocation List (CRL)** or via an **Online Certificate Status Protocol (OCSP)** responder.

So, as you can imagine, AD CS is a very critical asset and a high-value target for malicious use. Let's learn why it is important to use MDI on AD CS.

Importance of MDI on Certificate Servers

Placing MDI sensors on certificate servers is critical because it allows for the monitoring of certificate issuance and management activities, which can be targets for cyber threats. These sensors help detect anomalies in certificate requests and usage that may indicate malicious activity.

To illustrate the severity of an attack against AD CS, consider a scenario where a malicious actor exploits a misconfiguration in your AD CS server to obtain or forge a certificate. This certificate essentially acts as a password, allowing the attacker to bypass identity and access management controls and establish a presence within your organization – without the need to compromise any

actual passwords. This kind of breach underscores the critical importance of securing certificate management processes. Threat actors can exploit AD CS by targeting misconfigurations or weak policies to issue unauthorized certificates, enabling them to masquerade as legitimate users or systems. This can facilitate man-in-the-middle attacks, data breaches, and elevation of privilege. MDI sensors can help detect unusual patterns in certificate issuance or the abuse of certificate templates, contributing to early threat detection and response.

The following events are required for AD CS servers:

- **4870:** Certificate Services revoked a certificate
- **4882:** The security permissions for Certificate Services changed
- **4885:** The audit filter for Certificate Services changed
- **4887:** Certificate Services approved a certificate request and issued a certificate
- **4888:** Certificate Services denied a certificate request
- **4890:** The certificate manager settings for Certificate Services changed.
- **4896:** One or more rows have been deleted from the certificate database

We are now ready to configure our environment to fit the MDI deployment toward AD CS.

Configuring AD CS for MDI sensor installation

To be able to see all the events and get the alerts related to AD CS, we need to do some configuration.

If you haven't installed or imported the PowerShell module on the servers, please do so and follow the guide in [Chapter 3](#).

We only have one configuration that needs to be done on each AD CS server:

- AD CS auditing

OFFLINE ROOT CA AND MDI SENSOR

It's not required to install the MDI sensor on an offline Root CA. Offline Root CAs should remain isolated to ensure their security, only coming online for brief maintenance windows. Monitoring unexpected activity, such as unusual operational hours, is crucial and could indicate potential security threats. Instead, focus on deploying MDI sensors on active network servers where regular monitoring can provide more significant security insights.

AD CS advanced auditing

Advanced auditing in AD CS is crucial for maintaining the integrity and security of Certificate Services operations. By enabling detailed tracking of actions related to Certificate Services, administrators can enhance security, comply with regulatory requirements, and ensure operational integrity.

To start, auditing must be enabled in Windows to record the specific audit events selected in the CA properties to the security log. This capability is managed through **Advanced Audit Policy Configuration**, specifically under the **Object Access** category (we will configure this in the next section).

First, you must enable the audit of object access via **Group Policy**. This setting is crucial as it allows the security logging of any object access events, including those related to the CA.

Once the **Group Policy** setting is enabled, you can select which specific CA operations you want to audit by using the CA management console. The CA will then start logging the selected events, such as issuing certificates or changing CA configuration, to the Windows Security Log.

Enabling advanced auditing through Group Policy

Advanced auditing in AD CS starts with the enabling of object access audits via **Group Policy**. This setting is pivotal as it lays the groundwork for logging all relevant access events, particularly those pertaining to Certificate Services.

To enable **Group Policy** audit settings, follow these steps:

1. Navigate to and access the **Group Policy Management Console (GPMC)** on your domain controller.
2. Create or modify a **Group Policy Object (GPO)** that impacts your AD CS servers.
3. Within the GPO, go to **Computer Configuration > Policies > Windows Settings > Security Settings > Advanced Audit Policy Configuration > Audit Policies > Object Access**.
4. Enable auditing for **Audit Certification Services for Success and Failure** (see *Figure 4.6*).

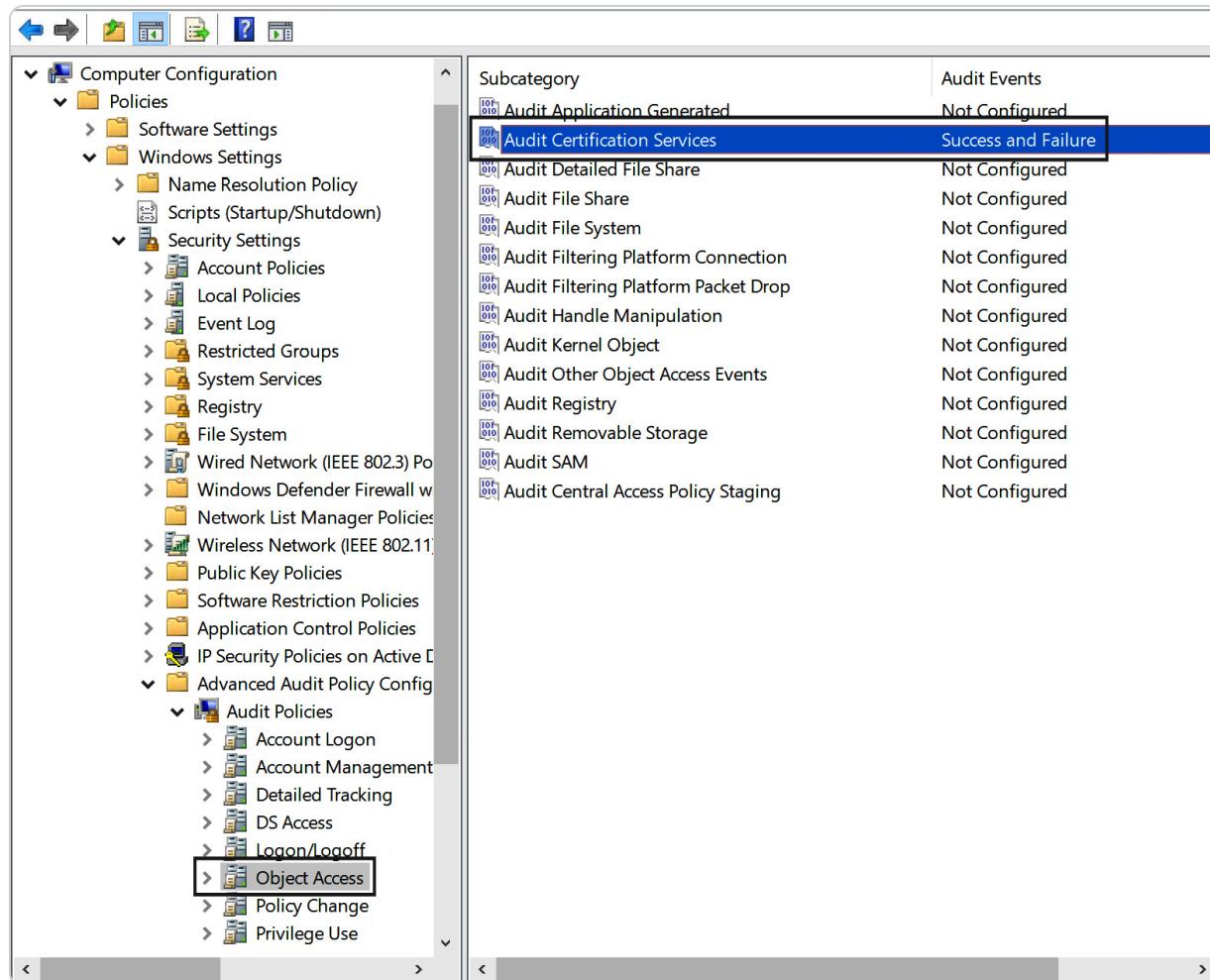


Figure 4.6 – GPO Audit Certification Services

Make sure you select both successful and unsuccessful access attempts. This level of inspection is critical for detecting potential security threats and ensuring compliance with the auditing requirements for AD CS.

Close the window and then apply the GPO to your AD CS servers.

Implementing CA auditing via PowerShell

The commands provided in this section will configure the audit logging level of Certificate Services, modify critical registry settings, and restart the service to apply these changes.

Follow these steps to configure the audit logging:

1. Start Windows PowerShell as an administrator to execute system-level modifications without restrictions.

NOTE

Make sure you run PowerShell as an administrator because modifying registry settings and stopping/starting services requires elevated permissions.

2. Type the following command to modify the CA audit settings:

```
certutil -setreg CA\AuditFilter 127
```

3. This command sets the audit filter to log all events. The **certutil** command is used to modify the registry setting for Certificate Services. The **-setreg** parameter specifies the registry key and the value to set. In our case, **CA\AuditFilter 127** sets the audit filter to log all events (since 127 is the sum of all event type values).

4. To ensure the new settings take effect immediately, cycle the Certificate Services:

```
Stop-Service -Name certsvc  
Start-Service -Name certsvc
```

5. This sequence gracefully stops and then restarts the service, minimizing downtime.

Configuring CA auditing using the GUI

For administrators who prefer a graphical interface over command-line tools, the CA management console offers an intuitive method to configure auditing.

Follow these steps to configure the audit logging:

1. Navigate to **Start > Certification Authority**.
2. Right-click your CA's name and select **Properties**.
3. Switch to the **Auditing** tab, and select all the events for auditing:
 - Backup and restore the **CA Database**
 - Change **CA Configuration**
 - Change **CA Security Settings**
 - Issue and manage **Certificate Request**
 - Revoke certificates and publish **CRLs**
 - Store and retrieve **archived keys**
 - Start and stop **Active Directory Certificate Services**
4. Click on **Apply**.

We have now successfully implemented the required auditing settings for the MDI sensor to be able to detect potential threats regarding AD CS. In the next section, we will verify that we have the correct implementation.

IMPORTANT NOTE

*Enabling the **Start and Stop Active Directory Certificate Services** event can lead to restart delays if you have a large AD CS database. You might choose not to enable this specific type of event auditing to avoid the potential for delays. If you are familiar with the AD CS*

database, you can also consider purging unnecessary entries from the database.

Validating the AD CS integration

During this validation step, we will generate test events to see whether they are sent to the MDI service. We will focus on changing one of the CA settings and also generate a certificate request, approve it, and then revoke the certificate.

To be able to verify that the events are sent to the MDI service, we need to do the following:

1. Verify that the sensor service is started and running using Windows PowerShell (or PowerShell 7 if you have installed that on the certificate server) by typing the following command:

```
Get-Service AATPSensor | Select-Object Status
```

The output of the preceding command should be `Running`.

2. We will now generate some test events to see whether the events are sent to MDI:

- Create a test certificate request with PowerShell using the following code:

```
$certRequest = @"
[Version]
Signature = "$Windows NT$"
[NewRequest]
Subject = "CN=TestUser, OU=YourOU, O=YourOrg, C=US"
KeyLength = 2048
Exportable = TRUE
MachineKeySet = FALSE
SMIME = FALSE
PrivateKeyArchive = FALSE
UserProtected = FALSE
UseExistingKeySet = FALSE
ProviderName = "Microsoft RSA SChannel Cryptographic Provider"
ProviderType = 12
RequestType = PKCS10
KeyUsage = 0xa0
"@

@"
$certRequest | Out-File -FilePath $requestFile
```

- Now, we will save the certificate request data to a file:

```
$requestFile = "C:\Temp\ESC1_certRequest.inf"
$certRequest | Out-File -FilePath $requestFile
```

- Generate the certificate request with the following:

```
$certReqFileOutput = "C:\Temp\ESC1_certReq.req"
certreq -new $requestFile $certReqFileOutput
```

- Open the file located at `C:\Temp\ESC1_certReq.req` with Notepad and copy its content to the clipboard.
- On your AD CS server, open Edge (or Internet Explorer if you have an older operating system) and go to <http://localhost/certsrv>.
- Click on **Request a certificate** (see *Figure 4. 7*).

Microsoft Active Directory Certificate Services -- Contoso Root CA

Welcome

Use this Web site to request a certificate for your Web browser, e-mail client, or other program. By using a certificate, you can verify your identity to people you communicate with over the Web, sign and encrypt messages, and, depending upon the type of certificate you request, perform other security tasks.

You can also use this Web site to download a certificate authority (CA) certificate, certificate chain, or certificate revocation list (CRL), or to view the status of a pending request.

For more information about Active Directory Certificate Services, see [Active Directory Certificate Services Documentation](#).

Select a task:

- [Request a certificate](#)
- [View the status of a pending certificate request](#)
- [Download a CA certificate, certificate chain, or CRL](#)

Figure 4.7 – AD CS request certificate

- On the next page, click on **Advanced Certificate Request** .
- Paste the request from the file that you copied earlier and choose the **ESC1** template (see *Figure 4.8*). Click on **Submit** and, on the next page, download the certificate.

Microsoft Active Directory Certificate Services -- Contoso Root CA

Submit a Certificate Request or Renewal Request

To submit a saved request to the CA, paste a base-64-encoded CMC or PKCS #10 certificate request or PKCS #7 renewal request generated by an external source (such as a Web server) in the Saved Request box.

Saved Request:

Base-64-encoded certificate request (CMC or PKCS #10 or PKCS #7):

```
jn73DbuD6FDy6ztgnzuT21ng9gtxt5leXPi2ZisNi
7K4x5hxe5WGfiYxXeYDmpr+VrJFBL4uM8Mz6wmUVa:
mFRi0kQn1YjNoKo4BnkyQIP17acUyaZT7IRuc2+OA:
umxx+05ImINVR37AFFQj0cGi13T6PzdWMbEGhg==

-----END NEW CERTIFICATE REQUEST-----
```

Certificate Template:

Additional Attributes:

Attributes:

Figure 4.8 – ESC1 certificate request

- Start the AD CS tool (CA) from **Server Manager** , go to **Issued Certificates** , and find the certificate you just requested.

- Right-click on the certificate, and select **All Tasks > Revoke Certificate**. Select dates and **Reason code**, then click on **Yes**. These actions should generate logs that can be monitored.

3. Simulate unauthorized activities to generate logs for Defender.

- To verify that the MDI sensor is actively monitoring and sending data, try simulating unauthorized access or configuration changes. For instance, you could disable one of the events for auditing in the CA. This test helps confirm that the MDI system detects and reports changes, ensuring it is functioning correctly and providing real-time security alerts:

- Navigate to **Start > Certification Authority**.
- Right-click your CA's name and select **Properties**.
- Switch to the **Auditing** tab and deselect one of the checkboxes.
- Click on **Apply**.

4. If not logged in already, please log in to security.microsoft.com and, from the menu blade, choose **Investigation & response > Hunting > Advanced hunting**.

5. Enter the following query in the **New query** tab:

```
IdentityDirectoryEvents | where Protocol =~ "Adcs"
```

You should now see some rows with information (see *Figure 4.9*), and if not, you must check and verify the settings that we did earlier in the *Configuring AD CS for MDI sensor installation section*.

The screenshot shows the Microsoft Advanced hunting interface. At the top, there are two tabs: 'New query*' and 'Run query'. The 'Run query' tab is selected, showing a query editor with the following LINQ-like query:

```
1 IdentityDirectoryEvents
2 | where Protocol =~ "Adcs"
3 | project TimeGenerated, ActionType, Protocol, Application
```

Below the query editor, there are three tabs: 'Getting started', 'Results' (which is selected), and 'Query history'. The results table has columns: 'TimeGenerated', 'ActionType', 'Protocol', and 'Application'. There are two items listed:

TimeGenerated	ActionType	Protocol	Application
Aug 27, 2024 10:4...	ADCS Certificate issuance	Adcs	Active Directory
Aug 27, 2024 10:5...	ADCS failed certificate issuance	Adcs	Active Directory

Figure 4.9 – Verifying data ingestion in Advanced hunting

- If you deselected an audit setting, you will get an incident in the Defender XDR portal (see *Figure 4.10* and *Figure 4.11*).

The screenshot shows the Microsoft Sentinel Incidents page. At the top, there are links for 'Alert service settings' and 'Email notification'. Below that, a section titled 'Most recent incidents and alerts' shows 3 incidents. A search bar allows searching by name or ID, with a filter set to '1 Day'. There are buttons for 'Export', 'Copy list link', and 'Customize columns'. A 'Save' button is also present. The main table lists one incident: 'Suspicious disable of audit filters of AD CS involving one user' (Incident ID 3374), which is marked as 'Medium' severity. The table includes columns for 'Incident name', 'Incident Id', 'Tags', and 'Severity'.

Figure 4.10 – Disabled audit filter in AD CS incident

2. Click on the incident name and you will see more details about the alert story.

The screenshot shows the Microsoft Sentinel Alert story page for the incident 'Suspicious disable of audit filters of AD CS involving one user'. The breadcrumb navigation shows 'Incidents > Suspicious disable of audit filters of AD CS involving one user > Suspicious disable of audit filters of AD CS'. The 'View alert details' section shows two entities: 'localAdmin' (Source Account) and '2 Hosts' (Destination Hosts). The 'Alert story' section contains a diagram illustrating the attack flow: 'localAdmin' (User) has disabled audit logs of 'Contoso Root CA' (Certificate Authority), which is connected to 'CS' (Client System). The 'Important information' section provides context about AD CS and lists modification details, noting that the 'IssueAndManageCertificateRequests' filter was removed.

Figure 4.11 – Alert story

Great. We are now auditing our CA environment. Please read the *Certified Pre-Owned: Abusing Active Directory Certificate Services* blog and whitepaper to learn more about AD CS. Next up, we will configure the newest feature of MDI when it comes to protecting our identities, and that's the support of Entra Connect.

Integrating MDI with Entra Connect

Bringing MDI together with Microsoft Entra Connect is all about stepping up your security game. It's like having a vigilant guard dog that watches over both your on-premises and cloud identities. By linking these two, you get a full-picture view that's crucial for spotting and stopping identity threats. This integration will help us in preventing, detecting, and responding to privilege escalation attacks and credential theft – commonly started against Entra Connect.

How Entra Connect works

Entra Connect (formerly known as Azure AD Connect/AAD Connect) is a tool from Microsoft to help bridge the gap between on-premises AD environments and Entra ID. Its primary role is to facilitate a seamless and secure synchronization of identity data, including users, groups, and other directory objects, between these two environments.

We will not go into depth about Entra Connect and how you can configure it with SSO, PHS, PTA, and Federation.

KEY FEATURES

Synchronization : Entra Connect synchronizes various identity attributes from on-premises AD to Entra ID, ensuring that user information is consistent and up to date across your organization's on-premises and cloud services.

SSO : By maintaining a synchronized environment, Entra Connect enables SSO, allowing users to access multiple applications and services with a single set of credentials, without the need for repeated logins.

PHS : This feature synchronizes a hash of a user's on-premises AD password with Entra ID, allowing for unified password management.

PTA : This allows login validation directly against the on-premises AD without storing passwords in the cloud.

Here's what the synchronization flow looks like:

- **Connectors** : Entra Connect uses connectors (formerly management agents) to facilitate communication with different data sources such as Active Directory. These connectors handle both the import and export of data, allowing for efficient synchronization without the need for on-site agents.
- **Attribute flow** : The synchronization involves bidirectional attribute flow between the connector space (a local representation of data) and the metaverse (a unified view of identity data). This process includes copying or transforming data based on defined synchronization rules.
- **Connector space and metaverse** : Each connected data source is represented in the connector space. The metaverse aggregates these multiple views into a consolidated identity profile. Changes are managed locally in the connector space and synchronized with the metaverse as per scheduled operations.
- **Provisioning** : When new objects are detected by an authoritative source, they are projected into the metaverse and can trigger the creation of new objects in the connector spaces of other connected systems. This operation, known as provisioning, is critical for maintaining updated and accurate identity data across all systems.

So, like all the other brothers and sisters in Tier 0, Entra Connect is a high-value target. Its central role in synchronizing identities between on-premises AD and Entra ID makes it a critical asset in network infrastructure. This positioning means that any compromise could have widespread repercussions,

potentially granting unauthorized access to a wide array of resources across both environments. Therefore, securing Entra Connect is paramount to safeguarding the entire identity management framework within an organization. Now, we will start configuring our environment so that we can protect Entra Connect.

Configuring Entra Connect for the MDI sensor

To be able to see the events and get the alerts related to Entra Connect, we need to do one configuration on each Entra Connect server:

- Entra Connect auditing

IMPORTANT

Make sure to install the MDI sensor on both the active and staging servers for Entra Connect.

Entra Connect advanced auditing

The following Entra Connect event is required for the MDI installation and configuration:

- **4624:** An account was successfully logged on

Follow the next steps to manually create a GPO to enable auditing on your Entra Connect servers:

1. Open **Group Policy Management** on your domain controller.
2. Navigate to the specific GPO you want to edit or right-click on **Group Policy Object** and then click on **New** :
 - If a new GPO: Name the GPO as the other MDI-related GPOs, such as **MDI - Advanced Audit Policy for Entra Connect**
3. Access the **Policy Settings** :
 - Go to **Computer Configuration > Policies > Windows Settings > Security Settings > Advanced Audit Policy Configuration > Audit Policies > Logon/Logoff**.
4. Edit the Audit Logon settings:
 - Select **Audit Logon** .
 - Configure the policy to log **Success and Failure** events to ensure the comprehensive monitoring of logon activities (see *Figure 4.12*).

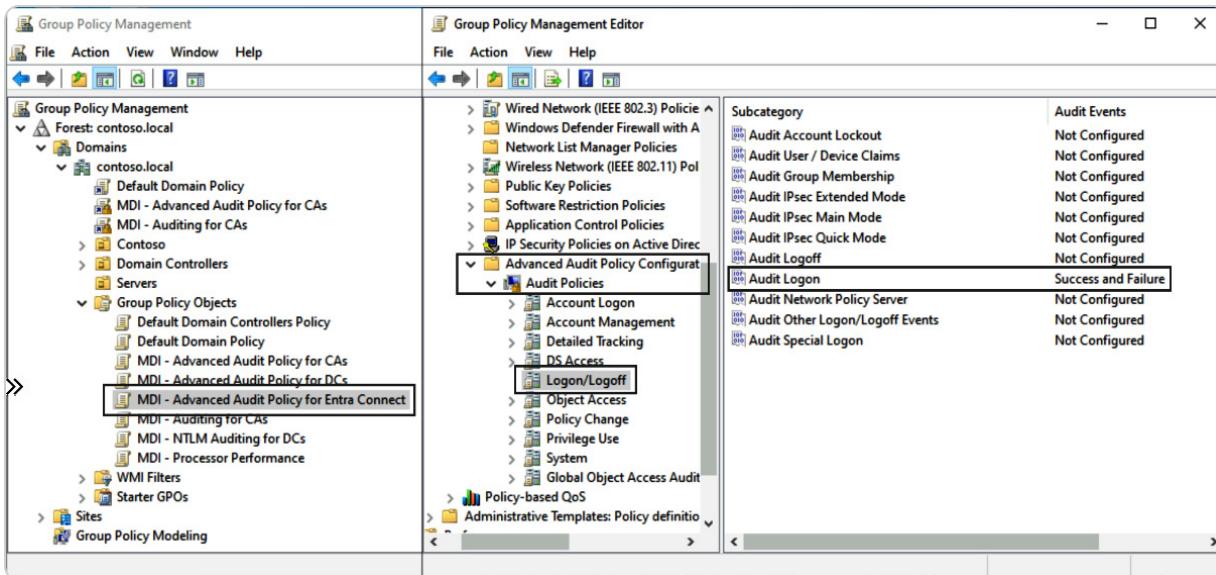


Figure 4.12 – Entra Connect advanced auditing GPO

5. Link the GPO and apply security filtering:

- After configuring the GPO, link it to the **Organizational Unit (OU)** that contains your Entra Connect servers.
- If necessary, based on your specific needs, implement **Security Filtering**. This restricts the GPO to only apply to your Entra Connect servers. To do this, modify the Security Filtering settings to include only the security groups or specific accounts associated with these servers.

This ensures that the GPO settings are precisely targeted and effective only where needed, enhancing both security and performance.

Now that we have the MDI sensor installed and the GPO created and linked to our Entra Connect servers, let's validate that the sensor works.

Validating the Entra Connect integration

During this validation phase, we ensure that everything is set up as planned. We'll also generate and monitor events to confirm that Entra Connect is properly sending event data to our **Hunting tables** in Defender XDR. Follow these steps:

1. Verify the sensor installation:

- Navigate to the **Sensors** settings page on the Defender XDR portal.
- Verify the server's name and sensor type. If the sensor type is marked as **Standalone**, this might suggest that the MDI sensor was installed before Entra Connect was configured. In this case, you should uninstall the MDI sensor, reboot the server, and then reinstall the MDI sensor to ensure proper integration.
- During the installation of the MDI sensor, if you find that all options are grayed out on the **Sensor deployment type** page, it indicates – as of the time this book was written – that the sensor type is set to **Entra Connect Sensor** (see *Figure 4.13*).

Microsoft Defender for Identity

The screenshot shows the Microsoft Defender for Identity interface. On the left, a sidebar titled 'General' has a 'Sensors' section selected. Below it are links for Activation, Directory services accounts, Manage action accounts, VPN, Adjust alerts thresholds, and About. Under 'Entity tags', there is a 'Sensitive' tag. The main content area is titled 'Deploying sensors enables you to monitor your on-premises Active Directory environment for [more](#)'. It includes 'Export' and 'Add sensor' buttons, and 'Filter' and 'Reset' options. A table lists four sensors: DC0 (Domain controller Sensor, contoso.local), ADFS0 (AD FS Sensor, contoso.local), CS (AD CS Sensor, contoso.local), and EntraConnect0 (Entra Connect Sensor, contoso.local). The 'EntraConnect0' row is highlighted with a red border.

Sensor	Type	Domain
DC0	Domain controller Sensor	contoso.local
ADFS0	AD FS Sensor	contoso.local
CS	AD CS Sensor	contoso.local
EntraConnect0	Entra Connect Sensor	contoso.local

Figure 4.13 – Entra Connect Sensor type

2. Verify the log:

- On the Entra Connect server, go to the `C:\Program Files\Azure Advanced Threat Protection Sensor\<version>\Logs` directory.
- Open the log file named `Microsoft.Tri.Sensor.Updater.log`.
- Search for the following specific string: `Warn SensorUpdaterConfiguration Updater WriteMachineName [MachineName=YourServerName, DomainController=False, Adfs=False, Adcs=False, AdConnect=True]`.
- Ensure that the log contains `AdConnect=True`. This confirms that the sensor is correctly configured to recognize and integrate with the Entra Connect setup.

3. Perform event generation and monitoring:

- From Entra ID and on a user that's synced from Active Directory – preferably a Domain Admin account – try to make a password reset. This will fail in a standard setup of Entra Connect but will generate a log in our Hunting table (`IdentityDirectoryEvents`).
- In the Defender XDR portal, access the **Advanced hunting** section and write the following query:

```
IdentityDirectoryEvents
```

- As you can see (*Figure 4.14*), we now have logs from Entra Connect.

The screenshot shows the Microsoft Advanced hunting interface. At the top, there's a navigation bar with 'New query*', a plus sign for creating new queries, and other options like 'Run query', 'Last 3 hours', 'Save', 'Share link', and 'Manage rules'. Below the navigation is a query editor window with the following content:

```

1 IdentityDirectoryEvents
2 | where Application =~ "Entra Connect"

```

Below the query editor, there are tabs for 'Getting started', 'Results' (which is selected), and 'Query history'. Under the results tab, there are buttons for 'Export', 'Show empty columns', and a search bar. To the right of the search bar are filters for 'TimeGenerated', 'Timestamp', 'ActionType', 'Application', 'TargetAccountUpn', and 'TargetAccountDisplayN'. A single item is listed in the results table:

TimeGenerated	Timestamp	ActionType	Application	TargetAccountUpn	TargetAccountDisplayN
Sep 4, 2024 10:07....	Sep 4, 2024 10:07:54 PM	Entra Connect failed password writeback	Entra Connect	baduser@contoso.local	baduser

Figure 4.14 – Entra Connect events in the IdentityDirectoryEvents table

Next up, let's learn how we can expand MDI across multiple AD forests.

Expanding MDI across multiple Active Directory forests

In today's complex and often fragmented enterprise environments, managing security across multiple OUs and geographical boundaries is not just a necessity but a mandate. Active Directory, Microsoft's directory service for Windows domain networks, is at the heart of identity management in most corporate environments. As organizations grow and evolve, they often find themselves managing not just one but multiple AD forests. Each forest can function as a distinct administrative, security, and policy boundary, raising unique challenges and opportunities in security management.

The concept of multiple forests

Active Directory forests are top-level containers in AD, which include one or more domains that are grouped together under a common schema and configuration. In a multi-forest environment, forests operate independently from one another, each with its own set of administrative policies and security settings. This architecture is typically employed for several reasons:

- **Organizational autonomy** : Different parts of a business may require autonomy and control over their IT resources, necessitating separate forests
- **Security isolation** : Critical business units or projects might demand isolation to enhance security, prevent breaches from propagating, and minimize the risk of insider threats
- **Mergers and acquisitions** : New forests may be created as companies merge or acquire others, integrating disparate IT systems while maintaining operational boundaries

Types of trusts in multi-forest environments

Trusts are foundational in a multi-forest architecture as they define how different forests recognize and share resources with each other. Understanding these trusts is crucial for security and interoperability:

- **Two-way trust** : A bidirectional agreement where each forest recognizes and trusts the other, allowing for resource sharing in both directions.
- **One-way trust** : A unidirectional relationship where one forest trusts another, but not vice versa. This setup is common when an organization needs to maintain strict control over resource access.
- **Transitive and non-transitive trusts** : Transitive trusts extend trust relationships beyond two forests, while non-transitive trusts do not. Managing these trusts requires careful planning to ensure security without overly complicating resource access.

Prerequisites for MDI in a multi-forest environment

To effectively monitor cross-forest activities, MDI sensors need to interact with domain controllers in various forests to gather detailed profiles of all relevant entities, including users and computers from those remote forests. This necessitates the deployment of MDI sensors on domain controllers across all forests, regardless of existing trust relationships.

To get started with setting up MDI sensors in a multi-forest environment, it's important to follow a straightforward deployment plan:

- **Naming conventions** : Implement a clear naming convention for the gMSAs across each forest and domain to avoid conflicts and enhance manageability.
- **Universal group setup** : Establish a universal group containing the computer accounts of all sensors. This setup facilitates access to the gMSAs' passwords and supports authentications across domains. Ensure that this group includes domain controllers and servers utilizing AD FS or AD CS from all participating forests and domains.

The universal group should have each of the domain controllers and AD FS/AD CS servers from the other forests and domains.

By using one universal group, we streamline management significantly. With one universal group, we obviously reduce the administrative burden, as we only need to manage memberships in one group rather than multiple groups spread across different domains. This simplification can reduce errors and inconsistencies in security policies.

Universal group considerations

When using universal groups, consider the following to ensure optimal performance and security:

- **Global catalog load** : Since universal group memberships are replicated to the global catalog, frequent changes to large universal groups can increase replication traffic and load on the global catalog servers. Plan and monitor accordingly.
- **Security** : Carefully manage who can modify universal group memberships to prevent unauthorized access to resources across your forest. Centralizing access control in one universal group also centralizes risk. If this group's management is compromised, the potential security breach could have wider impacts across all associated domains and forests. Proper security measures, such as strict access controls for this group and regular audits of its membership, are essential.

Kerberos ticket considerations

When a computer account is added to the universal group after it has already received its Kerberos ticket, it will be unable to retrieve the gMSA's password until a new Kerberos ticket is issued. This ticket reflects the group memberships as they were established at the time of its issuance.

NOTE

A new Kerberos ticket is required for updated group memberships to take effect. This ensures that the computer account can access necessary resources based on its current permissions.

To avoid disruptions, follow these steps:

- **Wait for the Kerberos ticket to renew:** Normally, tickets are valid for 10 hours before renewal.
- **Reboot the server:** Rebooting the server will force it to request a new Kerberos ticket that includes the updated group memberships.
- **Purge existing Kerberos tickets:** Execute the `klist purge -li 0x3e7` command from an administrator command prompt on the domain controller to manually clear existing tickets, thus ensuring a new one is issued that reflects the current group memberships.

Implementing these strategies will ensure that MDI sensors operate effectively across multiple forests, maintaining security and integrity.

Example design

In this section, we present a practical example to illustrate the deployment of MDI sensors in a multi-forest environment, featuring two forests: **ForestA** and **ForestB**. Each forest contains two domains, and we establish a two-way transitive trust between these forests to facilitate resource sharing and authentication.

Scenario overview

The goal with this type of scenario will be to install MDI sensors on all domain controllers in both forests and make them send data to one MDI workspace (security.microsoft.com):

- **Forests and domains :**
 - **ForestA** contains **DomainA1** and **DomainA2**
 - **ForestB** contains **DomainB1** and **DomainB2**
- **Trust relationship :** A two-way transitive trust is established between **ForestA** and **ForestB**, allowing resource sharing across forests

Deployment strategy in steps

1. Create a universal group in the forest root domain:

- **Action:** Create a universal group named **MDISensorsGroup** in **DomainA1**, which hosts the forest root domain for **ForestA**. Use the following PowerShell one-liner to **create** a new universal group. Make sure to adjust the **-Path** parameter to align with your organization's structure:

```
New-ADGroup -Name "MDISensorsGroup" -GroupScope Universal -GroupCategory Security -Path "OU=Groups,DC=DomainA1,DC=com"
```

- **Details:** The forest root domain acts as the central point for managing trust relationships and administrative control across the entire forest. Placing the universal group in the forest root ensures that group membership information can be replicated to the global catalog and shared across all child domains, facilitating seamless access control for MDI sensors in a multi-forest environment. This group will include all MDI sensor computer accounts from both forests and all domains.

2. Create naming convention for gMSAs:

- **Action:** Create gMSAs with distinct names reflecting their designated forest and domain, such as the following:
 - **gMSA-MDI-ForestA-DomainA1**
 - **gMSA-MDI-ForestA-DomainA2**
- **Details:** Having a separate gMSA per domain ensures that each domain can be managed independently, and permissions can be tightly scoped within each domain.

GMSA NAME UNIQUENESS

gMSA account names must be unique at the forest level, not just within a domain. Attempting to create a gMSA with a duplicate name, even across different domains in the same forest, will result in a failure. Ensure name uniqueness across all domains within the forest.

3. Assign permissions between gMSAs and the universal group:

- **Action:** Add **MDISensorsGroup** to the **PrincipalsAllowedToRetrieve ManagedPassword** property of each gMSA created, following this PowerShell one-liner:

```
Set-ADServiceAccount -Identity <gMSA name> ^  
-PrincipalsAllowedToRetrieveManagedPassword "MDISensorsGroup"
```

- **Details:** This setup ensures that the correct sensors in all domains can retrieve and use the gMSA credentials.

4. Deploy the sensor:

- **Action:** Install MDI sensors on domain controllers across all domains of both forests. Observe that MDI does support only one DSA to support all forests that have a two-way transient trust established.
- **Details:** Sensor configuration ensures they authenticate using the respective gMSA for their domain, facilitated by membership in **MDISensorsGroup**.

Configuration in steps

• Configure MDI sensors :

- **Action:** Log in to the Defender XDR portal and configure each sensor to use the corresponding gMSA based on its domain. This ensures that the sensors authenticate correctly in their respective domains.

- **Details:** Ensure that each gMSA has the proper permissions and configurations applied for smooth operation and deployment.

Security measures in steps

- Audit and monitor group membership :
 - **Action:** Perform regular audits on **MDISensorsGroup** .
 - **Details:** Ensure that no unauthorized accounts have been added to the group, and regularly monitor the usage of the gMSA accounts to detect any anomalies.

We have now configured the integration of the MDI sensors in a multi-forest AD environment. Don't forget to monitor changes and keep an eye on the Health Issues API (or the **Health Issues** page in the Defender XDR portal) for any misconfigurations. Next up, we will integrate MDI with a VPN.

VPN integration – securing remote activities and data flow

A lesser-known feature within MDI, and indeed across the Microsoft ecosystem, is the VPN solution provided by Microsoft. In today's remote work environment, robust security measures are essential to safeguard against evolving cyber threats. Integrating MDI with Microsoft RRAS through **Remote Authentication Dial-In User Service (RADIUS)** will be the focus of this section, but before we explore that, I want to highlight a new kid on the block – **Security Service Edge (SSE)**.

My perspective on VPNs is that they are a legacy product, and my suggestion is to replace them with SSE solutions such as **Microsoft Entra Private Access** because it aligns with a growing trend in cybersecurity. This shift also reflects the evolving needs of modern enterprises, especially in how they manage security in increasingly distributed environments. Here are some thoughts about the new way of working with SSE:

- **Zero-trust security model :** SSE solutions typically operate under the zero-trust principle of *Never trust, always verify* . This model is more suited to today's cloud-centric and mobile-first business environments, where users need to securely access applications from anywhere.
- **Granular access control :** Unlike traditional VPNs, which might grant broad network access once authenticated, SSE solutions such as Entra Private Access provide granular access controls. These systems can dynamically adjust permissions based on continuous assessment of risk factors, such as user location, device security posture, and current threat intelligence.
- **Reduced latency and improved performance :** By directing traffic through optimized paths and reducing the need to route all traffic through a central data center, SSE can offer better performance, which is particularly beneficial for cloud services and SaaS applications.
- **Simplified security infrastructure :** Integrating various security functions such as **secure web gateway (SWG)**, **cloud access security broker (CASB)**, and **zero-trust network access (ZTNA)** into a unified platform reduces complexity and can lower costs related to managing multiple separate security tools.

While the movement toward SSE solutions such as Microsoft Entra Private Access is well-justified for many modern enterprises, the decision to transition from VPNs should consider the specific needs and circumstances of the organization, including legacy systems (but they need to be replaced, as *legacy system* often means vulnerabilities), compliance requirements, and readiness for change. A phased approach that includes running both systems in parallel might be appropriate for some organizations, allowing them to gradually transition while ensuring continuous security and accessibility.

Understanding RRAS and RADIUS

While the Microsoft ecosystem offers a range of solutions for remote access and network security, not all these options are equally suited to modern enterprise needs. One such legacy solution is RRAS, which, when integrated with RADIUS, provides a traditional approach to managing remote user connectivity. However, in an era where SSE solutions are becoming the new standard, it's important to critically assess the limitations of RRAS and RADIUS, especially as enterprises transition toward more advanced security frameworks.

RRAS

RRAS has long been a staple in Microsoft's suite of network services, primarily known for its ability to configure VPNs, routing, and remote access. Despite its historical significance, RRAS is increasingly viewed as a legacy technology, with several limitations that are becoming more apparent in today's rapidly evolving IT landscape.

- **Traditional VPN connectivity :** The VPN functionality in RRAS, once considered a robust solution for secure remote access, is now showing its age. Traditional VPNs, such as those supported by RRAS, are increasingly seen as cumbersome, especially in environments where cloud-based applications and remote work are common. The *all-or-nothing* access model of VPNs, where users gain broad network access upon authentication, does not align well with the zero-trust security models that are now considered best practice.
- **Complex remote access management :** Managing remote connections through RRAS requires significant administrative overhead. The reliance on older authentication methods and the lack of integration with modern **identity and access management (IAM)** systems can lead to increased complexity and potential security risks, particularly in distributed work environments where flexibility and scalability are essential.

RADIUS

RADIUS plays a key role in centralizing **Authentication, Authorization, and Accounting (AAA)** processes but, like RRAS, it too shows its age in certain respects. While RADIUS continues to be a valuable tool for network access control, its traditional implementation can be a double-edged sword:

- **Rigid authorization controls :** The authorization mechanisms in RADIUS are often static and can lack the flexibility needed to support dynamic access controls. Unlike modern SSE solutions that offer granular, context-aware access, RADIUS typically operates within a more rigid framework. This can result in either over-permissive access, which increases security risks, or overly restrictive policies that hamper productivity.

- **Limited accounting capabilities** : While RADIUS does provide accounting functions, its logging and auditing capabilities are somewhat limited compared to the comprehensive, real-time analytics offered by contemporary security solutions. This can make it challenging for organizations to gain deep insights into user behavior, identify emerging threats, or ensure compliance with increasingly stringent regulatory requirements.

While RRAS and RADIUS have served organizations well in the past, their limitations are increasingly apparent in the context of today's security and networking needs. But let us go back in time and configure RRAS, integrate RADIUS, and configure MDI for RADIUS logs.

MDI VPN VENDOR SUPPORT

Please consult the vendor's documentation on how to enable RADIUS accounting.

Supported vendors include Check Point, Cisco ASA, F5, and Microsoft.

Configuring Microsoft RRAS

Let's install a new Windows server and join that server into the domain that you have. I haven't included any additional servers in the lab for this book, but if you have your own lab environment, please try out the upcoming steps. The RRAS feature is not supported in Azure.

Step 1 – Install RRAS

1. Log in with your administrative account.
2. Open **Server Manager** on your Windows server.
3. Click on **Manage** and select **Add Roles and Features**.
4. Proceed through the wizard until you reach the **Roles** section.
5. Check the box for **Remote Access** and then click on **Next** until you reach the **Role Services** section of the **Remote Access** role.
6. Select **Routing and Remote Access Services** and then finalize the installation.

Step 2 – Configure RRAS

1. After installation, open the **Routing and Remote Access** console from the **Tools** menu in **Server Manager**.
2. Right-click the server's name and select **Configure and Enable Routing and Remote Access**.
3. In the setup wizard, select **Custom configuration**.
4. Choose VPN access and NAT if needed for your network setup.
5. Complete the wizard and start the RRAS service.

Step 3 – Integrate RADIUS

1. In the **RRAS** console, right-click your server and select **Properties**.
2. Go to the **Security** tab.
3. Under **Authentication provider**, select **RADIUS Authentication** and click **Configure**.

4. Click on **Add** to enter the server's name of your MDI server and have port **1813** configured:
 - It is highly recommended to have some sort of high availability mindset, so add more MDI servers with the same settings

5. Enter a shared secret key (alphanumeric, which is a plaintext password used between a RADIUS client and server) by first clicking on **Change**. This secret will be used between the RRAS and RADIUS server. Make sure to record this secret as it will be needed on the RADIUS server side.

6. In the **Add RADIUS Server** dialog box, make sure to click on the checkbox saying **Send RADIUS Account On and Accounting Off messages**.

7. Click on **OK** on all dialog boxes.

Step 4 – Configure MDI to use RADIUS logs

1. If not already logged in to the Defender XDR portal (<https://security.microsoft.com>), please log in with your Security Administrator account.

2. Go to **System > Settings > Identities > VPN**.

3. Click on the **Enable RADIUS accounting** checkbox.

4. Enter the shared secret that you created in the previous step.

5. Click on **Save**.

6. Now, the MDI sensor will activate a Windows Firewall rule named **Microsoft Defender for Identity Sensor**, which allows incoming RADIUS accounting on port UDP/ **1813**.

Step 5 – Verify and monitor

When the Defender for Identity sensor captures VPN events and transmits them to the Defender for Identity cloud service for analysis, the entity profile highlights specific VPN locations accessed, and profile activities reflect these locations.

I say it again because it's worth mentioning it – everything is just not a box to tick. If we look at how easily we can misconfigure core and fundamental systems that we need to rely on, and how easily we can give away our keys to the kingdom... we need to make sure that we configure our environment in the correct way and monitor changes. Transitioning to newer, more modern, and secure solutions can help us all to be a few steps ahead of malicious actors.

RADIUS logs could be a valuable source for monitoring authentication attempts, helping detect suspicious activity such as failed logins, unusual access patterns, or privilege escalation attempts. They provide visibility into network access behavior, offering early indicators of potential identity compromise or lateral movement within your environment.

Now, let's review the main points we have discussed in this chapter.

Summary

In this chapter, we first looked at how we could integrate MDI together with AD FS and the necessary settings and permissions required. We then moved on to integration with AD CS and learned that many organizations need to verify their AD CS setup to improve their security posture. We also covered the newest sensor feature and installed the sensor on the Entra Connect server.

For organizations that have multi-forest and/or multi-domain setups, we learned how the MDI setup should look and the importance of protecting the universal group.

Finally, we learned how to implement RRAS together with RADIUS in MDI and also introduced the new SSE solution for future-proof remote work.

In the next chapter, we will delve into the world of the APIs that MDI can offer.

5

Extending MDI Capabilities Through APIs

In this chapter, we will explore how to extend the capabilities of Microsoft Defender for Identity (MDI) using the **Microsoft Graph API**. At the time of writing this book, in mid-2024, there is limited API support for MDI. We will focus on key APIs that allow you to monitor and manage alerts, incidents, and health issues within your MDI environment. By the end of this chapter, you will have a comprehensive understanding of how to use these APIs to enhance your security operations through automation and integration.

In this chapter, we're going to cover the following main topics:

- Introduction to the MDI API
- Building custom integrations and automations

Let's get started!

NEWS FROM MICROSOFT IGNITE 2024: SENSOR MANAGEMENT API FOR AUTOMATED OPERATIONS

To further enhance operational efficiency, Microsoft has launched a Sensor Management API. This API allows for the automation of tasks such as deployment, configuration, and monitoring of sensors within an organization's environment. By providing programmatic access, it enables security teams to maintain up-to-date sensor deployments and monitor their health status effectively, ensuring continuous and robust protection.

Technical requirements

Completion of [Chapter 2](#) is a prerequisite for this chapter and, as a reminder, you also require the following:

- Microsoft tenant
- Microsoft subscription that includes MDI, such as Microsoft 365 E5 or Microsoft 365 E3 + E5 Security
- Basic Microsoft 365 knowledge
- Basic PowerShell knowledge:
 - Windows PowerShell 5.1 or PowerShell 7.4 or later
 - Make sure your server has one of the above versions, or later, installed
- Postman (optional)
- Azure subscription (if you don't have an Azure subscription, create a free account before you begin)

- Bicep tools
- Visual Studio Code
- Azure CLI or Azure PowerShell module
- Ensure that you have Bicep version v0.27.1 or later for your Visual Studio Code extension and Azure CLI or Bicep CLI if you are using the Azure PowerShell module

All the code examples for this chapter can be found on GitHub at

<https://github.com/PacktPublishing/Microsoft-Defender-for-Identity-in-Depth/tree/main/Chapter05>

Introduction to the MDI API

In this section, you will get an overview of **Microsoft Graph**. We will discuss the MDI APIs, understand their significance, and learn how to get started with making API calls. This foundational knowledge is crucial for effectively utilizing the Graph API in subsequent sections. Understanding Microsoft Graph is essential for integrating various Microsoft services, including security solutions. By mastering these basics, you will be equipped to extend and automate your security operations and learn about the Graph API.

Why choose the Graph API? It's a gateway to a vast universe of Microsoft services, enabling you to access and manipulate a wide range of data across Microsoft 365 services and beyond. Using the Graph API allows for a unified, consistent platform for accessing rich, interconnected data sources, including user information, files, emails, and much more – all through a single access point. This means less juggling between different APIs and more streamlined, efficient development. For security applications such as MDI, it means more seamless integration and perhaps, in the future, quicker responses to potential threats.

Getting started with Microsoft Graph API

The **Microsoft Graph API** is like the central hub at a crowded airport, connecting flights (the data) from all over the Microsoft ecosystem, including Microsoft 365, Windows, and Azure. It streamlines how we interact with these services, making our digital work life a breeze by enabling us to create smarter, more integrated applications.

There is a distinction between an **app registration** and an **enterprise application** in **Entra ID** that is crucial to grasp, but before we examine this distinction, imagine that we have a **Logic App** in Azure that wants to make an HTTP request to the Graph API, which is quite common when we talk about automated responses from Microsoft Sentinel. Then, making a choice between an app registration and an enterprise application is necessary; do you want to use authorization via *headers* in the *HTTP*

request or use a *managed identity* (where we don't need to manage credentials) that can be enabled in the Logic App?

To extend this even more, developers and cloud people can keep their secrets safe in **Azure Key Vault**, but applications still need a way to access those secrets. Managed identities offer a solution by providing an automatically managed identity in Microsoft Entra ID for applications. This allows applications to authenticate and connect to resources that support Microsoft Entra authentication seamlessly. By using managed identities, applications can acquire Microsoft Entra tokens without needing to handle credentials manually. I recommend you use managed identities to authenticate any resource that supports Microsoft Entra authentication.

App registration in Entra ID involves creating and configuring an application's identity within Microsoft Entra ID. This process includes assigning unique identifiers (application ID and tenant ID), setting up authentication credentials (certificates or secrets), and defining the permissions required for the application to access Entra ID resources.

Here's a brief comparison of app registrations and enterprise applications and how the relationship works:

- **App registration** : Also known as an *application object* , this is like your app's blueprint that's used across all tenants – it's the global version. This object lays out the main features and settings that all instances of the app will share.
- **Enterprise application** : Also known as a *service principal object* , this is the local version of your app for a specific tenant. Think of it as a customized instance that comes from the blueprint. It gets its basic settings from the application object but operates independently within its own tenant, allowing your app to sign in and access secured resources there.

FURTHER LEARNING

To delve deeper into the relationship between application objects and service principals in Microsoft Entra ID, visit
<https://learn.microsoft.com/en-us/entra/identity-platform/app-objects-and-service-principals> .

As you plan your approach for communicating with the Graph API, you have two main options:

- App registration: We will need to save some IDs and secrets (make sure to save them in a safe location such as Azure Key Vault)
- Managed identity: Ideal if we support Microsoft Entra authentication

When your application needs to access or query data across multiple tenants - such as when using the healthIssues API endpoint in a multi-tenant environment - the configuration becomes more complex. This complexity arises because your application must authenticate and operate within tenants other than your own, each with its own security settings and permissions. To handle this scenario, setting up an app registration in multi-tenant mode is essential. This involves additional configurations to ensure your application can securely authenticate and function across different tenants. The following guide will walk you through enabling this capability using both app registrations and enterprise applications:

- 1. Register the application :** Begin by registering the application in your primary Microsoft Entra ID tenant. During this process, you'll need to specify the necessary API permissions to access the required resources. For instance, to access the Graph API's `healthIssues` endpoint, you would include the permission `SecurityIdentitiesHealth.Read.All` .
- 2. Service principals in other tenants :** To enable the application to operate across different tenants, establish service principals for each secondary tenant. This setup allows the application to access resources and perform actions that are specific to each tenant's permissions and policies.
- 3. Connect the application :** Retrieve the application ID from your primary tenant. You can then use tools such as the Azure CLI or Microsoft Graph PowerShell to create a new enterprise application linked to this globally unique application ID:

- Using the Azure CLI:

```
az ad sp create --id <Application Id>
```

- Using Microsoft Graph PowerShell:

```
Connect-MgGraph -Scopes "Application.ReadWrite.All"
New-MgServicePrincipal -AppId <Application Id>
```

Having explored the concepts of app registrations and enterprise applications, it's time to put that theory into practice. Let's walk through the process of actually creating an app registration using the Microsoft Entra portal, ensuring your application can securely access the Microsoft Graph API.

Creating an app registration using the Microsoft Entra portal

The first step in using the Graph API for your applications is to secure API access by creating an app registration. This process grants your application the permissions it needs to interact with Microsoft services effectively. Here's a step-by-step guide to setting this up through the Microsoft Entra portal:

1. Navigate to <https://entra.microsoft.com/> and log in using an account that has the necessary permissions, such as the **Cloud Application Administrator** role. This role allows you to create and manage all aspects of app registrations and enterprise apps, with the exception of app proxy. Note that this is a privileged role, so proceed with the appropriate administrative rights.
2. In the **Identity** menu, go to **Applications > App registrations** .
3. Click on **+ New registration** .
4. In the **Register an application wizard** (App Registration) type and choose the following:
 - **Name** : Type a descriptive name for the app registration
 - **Supported account types** : Choose **Accounts in this organizational directory only (<Your Tenant Name> only - Single tenant)**
 - **Redirect URI**: For Postman, we need to make the following choices:
 - **Select a platform** : Web
 - **URI** : <https://oauth.pstmn.io/v1/callback>
5. After the app registration is created, go to **Manage > API permissions** .
6. Click on **+ Add permission** .
7. Click on **Microsoft Graph** .

8. Click on **Application permissions**.

9. Start typing to search for the permissions below:

- **SecurityIdentitiesHealth.Read.All**
- **SecurityAlert.Read.All**

Add more permissions as you wish, but always think least privileged.

10. When you have selected the permissions, click on **Add permissions**.

11. Click on **Grant admin consent for <Your Tenant Name>**.

Verify that the status changes from **Not granted for <Your Tenant Name>** to **Granted for <Your Tenant Name>**.

12. Click on **Certificates & secrets**, then click on **Client secrets**.

13. Click on **+ New client secret**.

14. Type a descriptive name in the **Description** field.

15. Pick a period of your choice under **Expires**, such as 90 days (3 months).

16. Save the client secret.

17. Go back to **Overview** and note the **Application (client) ID** and **Directory (tenant) ID** because we will need those IDs in the next step (see *Figure 5.1*).

The screenshot shows the Microsoft Entra admin center interface. The top navigation bar includes 'Microsoft Entra admin cent...', a search bar, and various icons. The main title is 'MDI-Health-Issues-App-Registration'. On the left, there's a sidebar with icons for Home, App registrations, Overview, Quickstart, Integration assistant, Diagnose and solve problems, Manage (Branding & properties, Authentication, Certificates & secrets), and Preview features. The 'Overview' section is currently selected. The main content area shows the application's details under 'Essentials':

- Display name: [MDI-Health-Issues-App-Registration](#)
- Application (client) ID: b8b4f971-4106-4806-87a8-df5ad9e56daf
- Object ID: d4b7ae85-7ea2-4c18-aaea-f58cae2eb681
- Directory (tenant) ID: [redacted]
- Supported account types: [My organization only](#)

On the right side, there are links for Client credentials ([0 certificate, 1 secret](#)), Redirect URLs ([Add a Redirect URI](#)), Application ID URI ([Add an Application ID URI](#)), and Managed application in local directory ([MDI-Health-Issues-App-Registration](#)).

Figure 5.1 – App registration overview

If you feel like trying out the new Entra ID support in Bicep, Microsoft's Domain-Specific Language, we will do that next.

Creating an app registration using Bicep

As of the time of writing this book, Bicep support for Graph API (limited to Entra ID resources) is very new and currently in preview. Imagine needing to deploy an enterprise application or an app registration across multiple tenants – **Infrastructure as Code (IaC)** becomes invaluable here, allowing you to author the file once and deploy it multiple times. Embracing DevOps practices is crucial for maintaining a compliant and secure cloud environment. Let’s look at this new capability.

BICEP EXPERIMENTAL FEATURES

Since Bicep is in preview for Graph API, we need to activate experimental features with a configuration file named `bicepconfig.json`. You can learn more at <https://aka.ms/bicep/experimental-features>. Be aware that it is currently not supported to deploy Bicep files containing Microsoft Graph provider via Visual Studio Code – use Azure CLI or Azure PowerShell, also known as the AzPowerShell module, which also recommends PowerShell 7.2 or later.

Before we dive into the Bicep file and start authoring it, we need to create a configuration file to open for this new capability:

1. Start by creating a folder on your hard drive. For this example, I’m using `C:\temp\AppRegistration`.
2. In **Visual Studio Code**, go to **File > Open Folder** and locate the folder that you created. Click on **Select Folder**.
3. If you get the following question on the screen, **Do you trust the authors of the files in this folder?**, simply click **Yes, I trust the authors**.
4. We will now need two mandatory files, one holding all the resources we want and the other holding the necessary configuration to enable experimental features. In the folder structure on the left side within Visual Studio Code, create two new files by right-clicking on the folder name and choosing **New File**. Name the files accordingly:

- `main.bicep`
- `bicepconfig.json`

5. Open `bicepconfig.json` and make sure it contains the following code:

```
{  
  "experimentalFeaturesEnabled": {  
    "extensibility": true  
  }  
}
```

This snippet is used to enable experimental features in Bicep configurations, specifically focusing on the extensibility feature. It should be included in a `bicepconfig.json` file, which holds settings that govern Bicep deployments. The `experimentalFeaturesEnabled` property is crucial as it allows the activation of features still in development and not yet included in the stable release. These features are typically enabled for testing and gathering user feedback. Specifically, the extensibility setting in this context allows Bicep to deploy non- **Azure Resource Manager (ARM)** resources by using an `extension` model, such as deploying resources through the Microsoft Graph extension/provider.

DEPRECATING THE TERM PROVIDER IN FAVOR OF EXTENSION

The term **provider** in Bicep will be replaced with **extension** to enhance clarity and prevent confusion. **provider** often refers to Azure Resource Providers in ARM contexts, which could be misleading when discussing Bicep's extensibility features. Additionally, the change to **extension** prevents naming conflicts in the **Microsoft.Resource/deployments** API, which already uses **providers** for a different purpose, thus ensuring smoother integration and clearer documentation. Read more about the change here: <https://github.com/Azure/bicep/issues/14374>.

Let's continue with the steps.

6. Save the **bicepconfig.json** file.
7. Open **main.bicep** and make sure it contains the following:

```
extension microsoftGraph
@description('The name of the application to create.')
param appName string = 'GraphAPI-MDI-Test'
@description('The name of the secret to create.')
param secretName string = 'GraphAPI-MDI-Test-Secret'
@description('The end date and time for the secret.')
param EndDateTime string = '2029-12-31T23:59:59Z'
resource app 'Microsoft.Graph/applications@v1.0' = {
    uniqueName: appName
    displayName: appName
    signInAudience: 'AzureADMyOrg'
    passwordCredentials: [
        {
            displayName: secretName
            endDateTime: EndDateTime
        }
    ]
    requiredResourceAccess: [
        {
            resourceAppId: '00000003-0000-0000-c000-000000000000' // Microsoft Graph
            resourceAccess: [
                {
                    id: 'f8dc971-5d83-4ele-aa95-ef44611ad351' // SecurityIdentitiesHealth.Read.All
                    type: 'Role'
                }
                {
                    id: '45cc0394-e837-488b-a098-1918f48d186c' // SecurityIncident.Read.All
                    type: 'Role'
                }
                {
                    id: 'bf394140-e372-4bf9-a898-299cf7564e5' // SecurityEvents.Read.All
                    type: 'Role'
                }
                {
                    id: '472e4a4d-bb4a-4026-98d1-0b0d74cb74a5' // SecurityAlert.Read.All
                    type: 'Role'
                }
            ]
        }
    ]
}
resource sp 'Microsoft.Graph/servicePrincipals@v1.0' = {
    appId: app.appId
}
output appId string = app.appId
```

```
    output spId string = sp.id
    output secretValue string = app.passwordCredentials[0].secretText
```

8. This Bicep file will create an app registration and create the first secret all in one deployment. Edit the three different **param** entries to fit your naming convention and environment.
9. Before we try to deploy **main.bicep**, check if you have PowerShell 7.4 or later, the Azure CLI, or the Azure PowerShell module together with Bicep installed on your system. We will use **winget**, Microsoft's package manager for Windows, to install the software we need; be aware that **winget** is available by default in Windows 11.
10. Start Windows PowerShell if you don't have PowerShell 7. If you already have PowerShell 7 installed, skip this step. To install PowerShell 7, type the following command:

```
winget install --id Microsoft.PowerShell --source winget
```

Using the Azure CLI

To use the Azure CLI, follow these steps. If you want to use Azure PowerShell for deployment, skip to the following section:

1. Open the newly installed PowerShell 7 from the start menu as an administrator and then type the following to install the Azure CLI:

```
winget install -e --id Microsoft.AzureCLI
```

NOTE

The Azure CLI automatically installs the Bicep CLI whenever it detects that a Bicep command is needed, so there's no need to manually install the Bicep CLI.

2. If you're not logged in, run the following command to log in to Azure and follow the sign-in prompt:

```
az login
```

3. Choose the subscription that you want to use to create a new resource group to deploy the Bicep file.
4. Change the path in the console to the location where you have the **main.bicep** file.
5. Type the following to create a new resource group (choose the name of your resource group and Azure location, such as **westeurope** or **eastus**):

```
az group create --name rg-GraphAPI-test-001 --location westeurope
```

6. Run the following query for deployment:

```
$AppRegDeploy = az deployment group create --resource-group rg-GraphAPI-test-001
--template-file .\main.bicep
```

7. Now, in the **\$AppRegDeploy** variable, we have some JSON output that we need to handle to gather our newly created secret. Type this to convert the JSON result to a PowerShell object:

```
$AppRegDeployObj = $AppRegDeploy | ConvertFrom-Json
```

8. We can now extract the secret value, which is located at **properties > outputs > secretValue > value**:

```
$secretValue = $AppRegDeployObj.properties.outputs.secretValue.value
```

- With a new variable called `$secretValue`, we will now find our secret (see *Figure 5.2*):

```
$secretValue
```

```
[C:\] PS:> $AppRegDeployObj

id          : /subscriptions/[REDACTED]/resourceGroups/rg-GraphAPI-test-01/providers/Microsoft.Resources/deployments/main
location    :
name        : main
properties  : @{correlationId=e4335e6f-f470-4c09-9a3b-6d14d5a19502;
               debugSetting=;
               dependencies=System.Object[];
               duration=PT2.4783706S;
               error=;
               mode=Incremental;
               onErrorDeployment=;
               outputResources=System.Object[];
               outputs=;
               parameters=;
               parametersLink=;
               providers=System.Object[];
               provisioningState=Succeeded;
               templateHash=13687742336181027505;
               templateLink=;
               timestamp=2024-10-17 20:30:08;
               validatedResources=}
resourceGroup: rg-GraphAPI-test-01
tags        :
type        : Microsoft.Resources/deployments

[C:\] PS:> $secretValue = $AppRegDeployObj.properties.outputs.secretValue.value
[C:\] PS:> $secretValue
Bj~8Q~XK5TVyy13Njs9nc6C003_GcJoOLo_o4deu
[C:\] PS:>
```

Figure 5.2 – Output from the `secretValue` variable

Using Azure PowerShell

To use Azure PowerShell (the Az PowerShell module), follow these steps:

- Open the newly installed PowerShell 7 as an administrator from the start menu.
- To install Azure PowerShell, we need to set the PowerShell execution policy to remote signed by running the following:

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned ^
-Scope Process
```

- Install Azure PowerShell (the Az PowerShell module) by running the following:

```
Install-Module -Name Az -Scope CurrentUser -Repository PSGallery -Force
```

This will ensure you have the Azure PowerShell (the Az PowerShell module) installed and ready to use.

- Install the Bicep CLI from `winget`. Type the following:

```
winget install -e --id Microsoft.Bicep
```

5. If you're not logged in, run the following command to log in to Azure and follow the sign-in prompt:

```
Login-AzAccount
```

6. Choose the subscription that you want to use to create a new resource group to deploy the Bicep file.

7. Change the path in the console to the location where you have the `main.bicep` file.

8. Type the following to create a new resource group (choose the name of your resource group and Azure location, such as `westeurope` or `eastus`):

```
New-AzResourceGroup -Name rg-GraphAPI-test-001  
-Location eastus
```

9. Run the following for deployment:

```
$AppRegDeploy = New-AzResourceGroupDeployment -ResourceGroupName rg-GraphAPI-  
test-001 -TemplateFile .\main.bicep
```

10. Now, in the `$AppRegDeploy` variable, we have the output from `New-AzResourceGroupDeployment`. To get the value of the secret, we simply type the following:

```
$AppRegDeploy.Outputs.secretValue.Value
```

If you feel unsure about how to handle the IDs and secrets securely, we can use another alternative that we learned about earlier, and that's an enterprise application.

Creating an enterprise application

Just as with app registrations, choosing to use an enterprise application provides additional benefits. Enterprise applications are critical when you need to apply specific configurations and manage access within a certain tenant. This approach allows you to enforce detailed access control and customize how the application interacts with other services in your environment. It is particularly useful for managing complex scenarios where granular permissions, role assignments, and conditional access policies are required.

Here, you have a step-by-step guide on how to add the `SecurityIdentitiesHealth.Read.All` permission to the enterprise application:

1. Start PowerShell 7 from your start menu as administrator.

2. Install the Microsoft Graph PowerShell Module by running the following command:

```
Install-Module -Name Microsoft.Graph
```

3. Authenticate to Microsoft Graph. You will be redirected to a browser to complete the sign-in:

```
Connect-MgGraph -Scopes "Application.ReadWrite.All", "Directory.ReadWrite.All"
```

4. Before creating an enterprise application, you must first have an app registration that the enterprise application will be based on. In this case, I'm naming the app registration `MDIHealthIssues`:

```
$appRegistration = New-MgApplication -DisplayName "MDIHealthIssues"
```

5. Once the app registration is in place, create a service principal (enterprise application) for it:

```
$servicePrincipal = New-MgServicePrincipal -AppId $appRegistration.AppId
```

6. Find the Microsoft Graph service principal:

```
$graphPrincipal = Get-MgServicePrincipal -Filter "servicePrincipalType eq 'Application' and appId eq '00000003-0000-0000-c000-000000000000'"
```

7. Add the permission for `SecurityIdentitiesHealth.Read.All` :

```
$appRoleId = $graphPrincipal.AppRoles | Where-Object { $_.Value -eq "SecurityIdentitiesHealth.Read.All" -and $_.AllowedMemberTypes -contains "Application" } | Select-Object -ExpandProperty Id
```

8. Assign the permission to your application:

```
New-MgServicePrincipalAppRoleAssignedTo -ServicePrincipalId $servicePrincipal.Id -AppRoleId $appRoleId -PrincipalId $servicePrincipal.Id -ResourceId $graphPrincipal.Id
```

9. Verify that the enterprise application within Entra ID has the `SecurityIdentitiesHealth.Read.All` permission (see *Figure 5.3*):

The screenshot shows the Microsoft Entra ID portal interface. The URL in the address bar is `Home > Enterprise applications | All applications > MDIHealthIssues`. The main title is **MDIHealthIssues | Permissions**, with a note below it: "Enterprise Application". On the left, there's a sidebar with sections like **Self-service**, **Custom security attributes**, **Security** (which is expanded), **Conditional Access**, **Permissions** (which is selected and highlighted in grey), and **Token encryption**. Below that is the **Activity** section with links for **Sign-in logs**, **Usage & insights**, **Audit logs**, **Provisioning logs**, and **Access reviews**. At the bottom is the **Troubleshooting + Support** section. The main content area is titled **Permissions**. It says: "Below is the list of permissions that have been granted for your organization. As an administrator, you can grant permissions to this app on behalf of all users (delegated permissions). You can also grant permissions directly to this app (app permissions)." There's a link to "Learn more". Below that, it says: "You can review, revoke, and restore permissions. [Learn more](#)". There's another link to "Configure requested permissions for apps you own, use the [app registration](#)". A button labeled "Grant admin consent for Thoor Security" is visible. Under the "Admin consent" tab, there's a table with one row for "Microsoft Graph". The columns are: API Name (Microsoft Graph), Claim value (SecurityIdentitiesHealth.Read.All), Permission (Read all identity security h...), Type (Application), and Grant Consent (Ad).

API Name	Claim value	Permission	Type	Grant Consent
Microsoft Graph	SecurityIdentitiesHealth.Read.All	Read all identity security h...	Application	Ad

Figure 5.3 – View of enterprise application with Graph API permission

The first step is done with these applications and the foundation is prepared for our future API calls.

Getting the access token

Next, we need to learn how to get an access token to communicate with the API. We will use OAuth 2.0 to acquire an access token.

Using Postman

Here is an example of how to obtain an access token using Postman:

1. Install or start Postman on your computer.
2. Change the method to **POST**.
3. Have your directory (tenant) ID ready in the **Enter URL or paste text** field and type the following command (change **<YourTenantID>** to the GUID you saved earlier):

```
https://login.microsoftonline.com/<YourTenantID>/oauth2/v2.0/token
```

4. Click on the **Body** tab within the request window.
5. From the dropdown, choose **x-www-form-urlencoded**.
6. Then, under **Key** and **Value**, type the following:

Key	Value
<code>grant_type</code>	<code>client_credentials</code>
<code>client_id</code>	Paste the application (client) ID
<code>client_secret</code>	Paste the client secret value
<code>scope</code>	<code>https://graph.microsoft.com/.default</code>

Table 5.1 - Postman body data

7. Now, we are ready to send this POST request to get the access token. You should get a **200 OK** (successful HTTP request) response with the following body:

```
{
  "token_type": "Bearer",
  "expires_in": 3599,
  "ext_expires_in": 3599,
  "access_token":
  "eyJ0eXAiOiJKV1QiLCJub25jZSI6ImhZNG45NERYT014S0Radng2WktEV3ZTQ0
  M0UEdOVDlxakV3NGM3Q0RtWDAlLCJhbGciOiJSUzI1NiIsIng1dCI6IkwxS2ZLRk
  lfam5YYndXYzIyeFp4dzFzVUhIMCIsImtpZCI6IkwxS2ZLRklfam5YYndXYzIye
  Fp4dzFzVUhIMCJ9..."
```

Now, save the long string value that you see under the `access_token` property. We will need to use this security token in the step where we will make an API call.

Using PowerShell

You know how to retrieve an access token from within the Postman application, but now we will do it with code and with PowerShell. We will use the same app registration that we created earlier, and you

need to have your tenant ID, client ID, and the client secret ready. But before we dive into getting an access token in PowerShell, I want you to explain the differences between two cmdlets – `Invoke-RestMethod` and `Invoke-WebRequest`.

`Invoke-RestMethod` and `Invoke-WebRequest` are both used in PowerShell to interact with web services, but they serve slightly different purposes and have different features:

- `Invoke-RestMethod` :

- **Purpose** : Specifically designed to interact with RESTful APIs.
- **Response handling** : Automatically parses the response based on the content type. For example, it converts JSON responses directly into PowerShell objects, which makes it easier to work with the data.
- **Simplicity** : Generally simpler to use for API calls because it abstracts away some of the complexity involved in handling web requests and responses.
- **Usage** : Ideal for retrieving data directly as objects. For example, to fetch user profile information from Microsoft Graph API using a GET request, you would use the following:

```
$response = Invoke-RestMethod -Uri "https://graph.microsoft.com/v1.0/me" -  
Headers $headers
```

- **Output** : Typically outputs the response content in a form that's easy to work with (e.g., objects for JSON data).

- `Invoke-WebRequest` :

- **Purpose** : More general-purpose web request tool.
- **Response Handling** : Does not automatically parse the response body. Instead, it returns a detailed `HttpWebResponseObject`, which includes properties for headers, content, status code, status description, and more.
- **Flexibility** : Provides more control over the web request and is useful when you need to handle the full details of the HTTP response or when working with non-API web resources (such as web scraping).
- **Usage** : Best for scenarios requiring detailed HTTP response handling. For instance, to obtain a complete response from the Microsoft Graph API with all the HTTP details, you would execute the following:

```
$response = Invoke-WebRequest -Uri "https://graph.microsoft.com/v1.0/me" -  
Headers $headers
```

- **Output** : Outputs a detailed response object that includes raw content, headers, status code, and more.

Let's look at an example with the Microsoft Graph API.

This PowerShell script demonstrates how to utilize `Invoke-RestMethod` to interact with the Microsoft Graph API. The example fetches user profile information by making a `GET` request. Here's how you can structure your request:

```
$uri = "https://graph.microsoft.com/v1.0/me"  
$headers = @{  
    "Authorization" = "Bearer $accessToken"  
}
```

```
$response = Invoke-RestMethod -Uri $uri -Headers $headers
$response
```

The output from the `$response` variable will look like this:

```
@odata.context      : https://graph.microsoft.com/v1.0/$metadata#users/$entity
businessPhones    : {}
displayName        : John Doe
givenName          : John
jobTitle           :
mail               : john.doe@contoso.com
mobilePhone        :
officeLocation     :
preferredLanguage  : en-US
surname            : Doe
userPrincipalName : john.doe@contoso.com
id                : 7706bdbf-09d6-4541-852b-28d076f8dd19
```

For scenarios requiring detailed HTTP response data, `Invoke-WebRequest` is particularly useful.

Here's how to set up and execute a request that provides comprehensive response details:

```
$uri = "https://graph.microsoft.com/v1.0/me"
$headers = @{
    "Authorization" = "Bearer $accessToken"
}
$response = Invoke-WebRequest -Uri $uri -Headers $headers
$response
```

Here's the output from the `$response` variable:

```
StatusCode       : 200
StatusDescription: OK
Content          : {"@odata.context": "https://graph.microsoft.com/v1.0/$metadata#users/$entity", "businessPhones": [], "displayName": "John Doe", "givenName": null, "jobTitle": null, "mail": "john.doe@contoso.com", "mobilePhone": ...}
RawContent       : HTTP/1.1 200 OK
                  Transfer-Encoding: chunked
                  Strict-Transport-Security:
                  max-age=31536000
                  request-id:
                  7706bdbf-09d6-4541-852b-28d076f8dd19
                  client-request-id:
                  7706bdbf-09d6-4541-852b-28d076f8dd19
                  x-m...
Forms            : {}
Headers          : {[Transfer-Encoding, chunked], [Strict-Transport-Security, max-age=31536000], [request-id, 7706bdbf-09d6-4541-852b-28d076f8dd19], [client-request-id, 7706bdbf-09d6-4541-852b-28d076f8dd19]...}}
Images           : {}
InputFields      : {}
Links            : {}
ParsedHtml       : mshtml.HTMLDocumentClass
RawContentLength: 354
```

Additionally, you can parse the JSON content from the response body to manipulate it further:

```
$data = $response.Content | ConvertFrom-Json
$data
```

Here's the output of the `$ data` variable:

```
@odata.context      : https://graph.microsoft.com/v1.0/$met
                      adata#users/$entity
businessPhones     : {}
displayName        : John Doe
givenName          : John
jobTitle           :
mail               : john.doe@contoso.com
mobilePhone        :
officeLocation     :
preferredLanguage  : en-US
surname            : Doe
userPrincipalName : john.doe@contoso.com
id                 : 7706bdbf-09d6-4541-852b-28d076f8dd19
```

Which one is easiest to use for the Microsoft Graph API? While both methods can be used, `Invoke-RestMethod` is more straightforward for typical API calls due to its automatic handling of JSON responses and simpler syntax.

This is because of the following reasons:

- It simplifies the process by automatically parsing JSON responses into PowerShell objects
- It allows you to easily work with the returned data without additional parsing
- It abstracts some of the complexities of making web requests, which is especially useful for straightforward API interactions

Let's get the access token with `Invoke-RestMethod` :

1. Start Windows PowerShell, or PowerShell 7 if it is installed on your computer.
2. Copy and paste or type the following code and insert the different IDs and the secret in the respective variables:

```
$tenantId = "<your_tenant_id>"
$clientId = "<your_client_id>"
$clientSecret = "<your_client_secret>"
$body = @{
    grant_type      = "client_credentials"
    scope           = "https://graph.microsoft.com/.default"
    client_id       = $clientId
    client_secret   = $clientSecret
}
$response = Invoke-RestMethod -Method Post -Uri
"https://login.microsoftonline.com/$tenantId/oauth2/v2.0/token" -ContentType
"application/x-www-form-urlencoded" -Body $body
$accessToken = $response.access_token
$headers = @{
    Authorization = "Bearer $accessToken"
}
```

Here's a detailed breakdown of the code:

- We started by replacing `<your_tenant_id>` and `<your_client_id>` with the IDs that you retrieved earlier. The client secret will also be inserted, and yes, this is not best practice to have such a secret directly within the script, but for now we will do it this way.

- We then construct our body with `grant_type`, `scope`, and then our `client_id` and `client_secret`.
- When we have those two parts ready, we can make a request and save the response in our variable that we named `$response`. In this request, we are using `Invoke-RestMethod` against the `Uri` that will get us the access token.
- In `$response.access_token`, we now have our `access_token`, which can be used later to talk to the Graph API endpoints. In this case, we are saving the `$response.access_token` to a new variable that we call `$accessToken`.
- In the last rows, you see that we are preparing the header with a new variable with the same name, `$headers`, that includes the header field name, `Authorization`, with the type of token, which in this case is the `Bearer` type, and we use the string of the `$accessToken` variable to form the entire header:
 - **Authorization header**: The `Authorization` header passes authentication information to the server, indicating who you are and what kind of access you have.
 - **Bearer token**: In OAuth 2.0, a bearer token signifies that the token holder is authorized to access the resource.

3. To view the output and the content of the `$headers` variable, simply type in `$headers` and press the *Enter* key. The output should look like the following:



```
[C:\] $ tenantId = "REDACTED"
[C:\] $ clientId = "345bafab-ae64-450e-8c33-a7edc7292392"
[C:\] $ clientSecret = "REDACTED"
[C:\] $ 
[C:\] $ body = @{
>>   grant_type    = "client_credentials"
>>   scope         = "https://graph.microsoft.com/.default"
>>   client_id     = $clientId
>>   client_secret = $clientSecret
>> }
[C:\] $ 
[C:\] $ response = Invoke-RestMethod -Method Post -Uri "https://login.microsoftonline.com/$tenantId/oauth2/v2.0/token" -ContentType "application/x-www-form-urlencoded" -Body $body
[C:\] $ accessToken = $response.access_token
[C:\] $ 
[C:\] $ headers = @{
>>   Authorization = "Bearer $accessToken"
>> }
[C:\] $ $headers

Name          Value
----          -----
Authorization Bearer eyJ0eXAiOiJKV1QiLCJub25jZSI6IlgyTzRPRk15eTI0dWg2ZE1fazhYNUI0dmRNkZ5LV9KY31KcHpX...
```

Figure 5.4 – Headers output with the bearer token value

We are now fully equipped to make API calls, but before that, we need to address some security best practices.

Security and compliance

When using APIs, especially in a security context, adhering to best practices for security and compliance is critical. Using the Microsoft Graph API requires careful consideration of various security measures to ensure the protection of sensitive data and compliance with relevant regulations. For a comprehensive overview of integrating with Microsoft's identity platform, refer to <https://learn.microsoft.com/en-us/entra/identity-platform/identity-platform-integration-checklist>. Here, I've expanded on some key aspects of this checklist to secure and manage your API keys and secrets effectively:

- **Secure API keys and secrets :**
 - **Storage** : Store credentials securely, using environment variables or secure vaults such as Azure Key Vault. Don't hardcode them in your scripts or applications.
 - **Rotation** : Regularly rotate API keys and secrets to minimize the risk of exposure.
 - **Access to keys and secrets** : Limit access to API keys and secrets to only those who need it.
- **Access control policies :**
 - **Principle of least privilege** : Implement strict access controls to ensure only authorized applications and users can interact with the APIs. Regularly review and update permissions to follow the principle of least privilege.
 - **Conditional Access policies** : Use Entra ID Conditional Access policies to enforce additional access controls based on user location, device status, or risk level.
- **Authentication :**
 - **OAuth 2.0 and OpenID Connect** : Use OAuth 2.0 for authentication and OpenID Connect for authentication and obtaining user information securely. Ensure your applications follow the recommended practices for securely handling tokens.
 - **Multi-Factor Authentication (MFA)** : Enforce MFA for accessing sensitive operations within your API to add an additional layer of security.
- **Secure data transmission :**
 - **HTTPS** : Always use HTTPS to encrypt data in transit between your application and the Microsoft Graph API. This protects against man-in-the-middle attacks.
 - **TLS configuration** : Ensure your servers are configured to use strong TLS protocols and ciphers.
- **Logging and monitoring :**
 - **Activity logs** : Maintain logs of API interactions for auditing and compliance purposes. These logs should include information on who accessed the API, what operations were performed, and when.
 - **Monitoring** : Use Azure Monitor, Microsoft Sentinel, or other **security information and event management (SIEM)** solutions to monitor API usage and detect any unusual activity.
- **Compliance considerations :**

- **Regulatory requirements** : Ensure that your use of the APIs complies with relevant regulatory requirements, such as the GDPR, HIPAA, or CCPA. This involves implementing data protection measures and maintaining data privacy.
 - **Data residency** : Be aware of the data residency requirements for the data you process and store. Use region-specific endpoints where necessary to comply with local data residency laws.
 - **Documentation and policies** : Keep comprehensive documentation of your API integration, including security policies and procedures. This documentation should be readily accessible to your development and security teams.
- **Security testing** :
- **Penetration testing** : Regularly conduct penetration testing to identify and address security vulnerabilities in your API implementation.
 - **Static and dynamic analysis** : Use static and dynamic analysis tools to scan your code for security vulnerabilities during development and runtime.

Making API calls

We will continue using PowerShell as our tool for making the API calls.

With the `access_token` string ready within our `$headers` variable, we can now make our first API call against the Microsoft Graph API.

Please note that if you have jumped directly into this section in the book, just follow the two steps in the following script. This example requires an existing app registration with appropriate permissions set up. Additionally, you must have the application (client) ID and client secret ready, as they are essential for making API calls. Ensure these prerequisites are met to successfully execute the upcoming script. The first step is to obtain the access token using the following script:

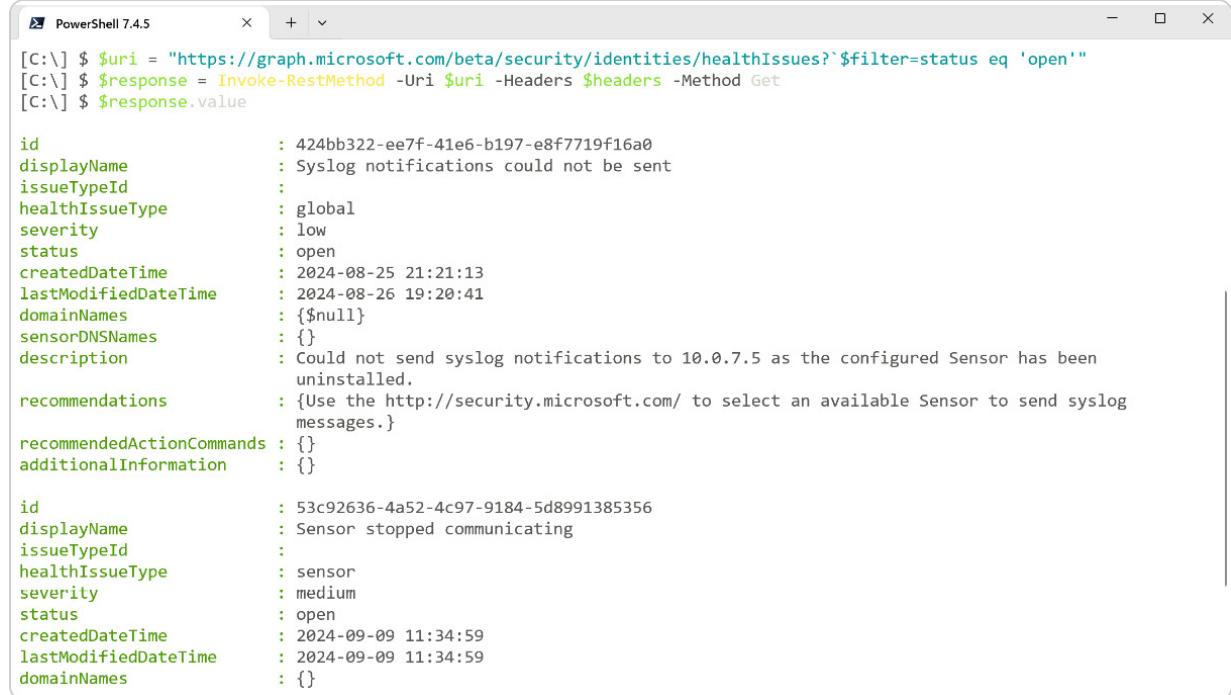
```
$tenantId = "<your_tenant_id>"  
$clientId = "<your_client_id>"  
$clientSecret = "<your_client_secret>"  
$body = @{  
    grant_type      = "client_credentials"  
    scope          = "https://graph.microsoft.com/.default"  
    client_id      = $clientId  
    client_secret   = $clientSecret  
}  
$tokenResponse = Invoke-RestMethod -Method Post -Uri  
"https://login.microsoftonline.com/$tenantId/oauth2/v2.0/token" -ContentType  
"application/x-www-form-urlencoded" -Body $body  
$accessToken = $tokenResponse.access_token  
$headers = @{  
    Authorization = "Bearer $accessToken"  
}
```

Next, we'll query the `healthIssue` endpoint to gather open issues with our sensors using the following URL (note the backtick ` character before the `$filter` ; that's needed in PowerShell for the filtering to work):

```
$uri = "https://graph.microsoft.com/beta/security/identities/healthIssues?  
`$filter=status eq 'open'"  
$response = Invoke-RestMethod -Uri $uri -Headers $headers -Method Get
```

We'll output the response using the following command (see *Figure 5.5*):

```
$response.value
```



The screenshot shows a PowerShell window titled "PowerShell 7.4.5". The command entered is:

```
[C:\] $uri = "https://graph.microsoft.com/beta/security/identities/healthIssues?$filter=status eq 'open'"  
[C:\] $response = Invoke-RestMethod -Uri $uri -Headers $headers -Method Get  
[C:\] $response.value
```

The output displays two objects, each representing a health issue. The properties and their values are:

Property	Value
id	424bb322-ee7f-41e6-b197-e8f7719f16a0
displayName	Syslog notifications could not be sent
issueTypeId	
healthIssueType	global
severity	low
status	open
createdDateTime	2024-08-25 21:21:13
lastModifiedDateTime	2024-08-26 19:20:41
domainNames	{\$null}
sensorDNSNames	{}
description	Could not send syslog notifications to 10.0.7.5 as the configured Sensor has been uninstalled.
recommendations	{Use the http://security.microsoft.com/ to select an available Sensor to send syslog messages.}
recommendedActionCommands	{}
additionalInformation	{}
id	53c92636-4a52-4c97-9184-5d8991385356
displayName	Sensor stopped communicating
issueTypeId	
healthIssueType	sensor
severity	medium
status	open
createdDateTime	2024-09-09 11:34:59
lastModifiedDateTime	2024-09-09 11:34:59
domainNames	{}

Figure 5.5 – Output from the open issues API call

Working with the `healthIssue` API endpoint

At the time of writing, the `healthIssue` API endpoint is still in its beta phase and is available through the beta endpoint. This indicates that it may undergo changes before it stabilizes and moves to the v1.0 endpoint.

The `healthIssue` API endpoint is designed to identify potential issues within a customer's MDI configuration, pinpointing concerns detected by MDI.

The following methods are supported:

- **List**: Retrieve a list of `healthIssues` objects (note the plural form) along with their properties.

Here's an example script:

```
$uri = "https://graph.microsoft.com/beta/security/identities/healthIssues"  
$response = Invoke-RestMethod -Uri $uri -Headers $headers -Method Get  
$response
```

In the `$uri` variable, we specify the URL of the API endpoint that retrieves a list of `healthIssues` objects, including their properties. The `$response` variable then captures the output from this API call, storing the returned data from the server, which includes details about each health issue.

Here's the output:

```
id : 5797eef0-a65a-4e4d-aa02-2bf42
     1d2ff9c
displayName : Sensor stopped communicating
issueTypeId :
healthIssueType : sensor
severity : medium
status : closed
createdDateTime : 2024-05-08T07:42:17.3051251Z
lastModifiedDateTime : 2024-05-08T08:06:13.2267217Z
domainNames : {}
sensorDNSNames : {DC01.contoso.local}
description : There has not been
               communication from the
               Sensor DC01.contoso.local
               for 05/08/2024 07:30 -
               05/08/2024 08:06. Last
               communication was on
               05/08/2024 07:30.
recommendations : {Check that the Sensor
                   service is up and running.,
                   Check the communication
                   between the Sensor to thoorse
                   csensorapi.atp.azure.com:443.
                   }
recommendedActionCommands : {}
additionalInformation : {}
```

We can now work with the output and, for example, summarize and group **healthIssueType** and **status**:

```
$summary = $response.value | Group-Object -Property healthIssueType, status |
ForEach-Object {
    [PSCustomObject]@{
        HealthIssueType = $_.Group[0].healthIssueType
        Status = $_.Group[0].status
        Count = $_.Count
    }
}
# Output the summary
$summary | Format-Table -AutoSize
```

The output from the last line with **\$summary** will then look like this:

HealthIssueType	Status	Count
global	open	2
global	closed	3
sensor	closed	15
sensor	open	2

- **Get**: This allows you to access the properties and relationships of a **healthIssue** object.

Here's an example script :

```
# Define the variables
$accessToken = "<access_token>"
$healthIssueId = "<id>"
$uri =
"https://graph.microsoft.com/beta/security/identities/healthIssues/$healthIssueId"
#
# Define the request headers
$headers = @{
    "Authorization" = "Bearer $accessToken"
```

```

    }
# Perform the GET request
$response = Invoke-RestMethod -Uri $uri -Method Get -Headers $headers
# Output the response
$response

```

Here's the breakdown:

- Define variables:
 - **\$accessToken** : Your access token
 - **\$healthIssueId** : The ID of the health issue you want to retrieve
 - **\$uri** : The endpoint URL with the health issue ID included
- Request headers: Set the **Authorization** header with the **Bearer** token.
- Perform the **GET** request:
 - Use **Invoke-RestMethod** to send the **GET** request with the defined URI and headers
 - Store and output the response
- **Update** : Modify the properties of a **healthIssue** object.

Here's an example script:

```

# Define the variables
$accessToken = "<access_token>"
$healthIssueId = "<id>"
$uri =
"https://graph.microsoft.com/beta/security/identities/healthIssues/$healthIssueId"
# Define the request headers
$headers = @{
    "Authorization" = "Bearer $accessToken"
    "Content-Type" = "application/json"
}
# Define the request body (example below)
$body = @{
    status = "resolved"
    additionalDetails = "Issue has been resolved."
}
# Convert the body to JSON
$jsonBody = $body | ConvertTo-Json
# Perform the PATCH request
$response = Invoke-RestMethod -Uri $uri -Method Patch -Headers $headers -Body
$jsonBody
# Output the response
$response

```

Here's the breakdown:

- Define variables:
 - **\$accessToken** : Your access token
 - **\$healthIssueId** : The ID of the health issue you want to update
 - **\$uri** : The endpoint URL with the health issue ID included

- Request headers:
 - Set the **Authorization** header with the **Bearer** token
 - Set the **Content-Type** header to **application/json**
- Request body:
 - Create a hashtable with the properties to update (**status** and **additionalDetails**)
 - Convert the hashtable to a JSON string using **ConvertTo-Json** cmdlet
- Perform the **PATCH** request:
 - Use **Invoke-RestMethod** to send the **PATCH** request with the defined URI, headers, and JSON body
 - Store and output the response

Working with the Alert API endpoint

In this section, you will learn how to use the Alert API endpoint, which is accessible via `/security/alerts_v2`, to manage security alerts. This knowledge will help you identify and respond to potential security threats more effectively. Let's explore how to practically apply the Alert API endpoint for enhanced security operations. These examples will guide you through filtering specific alerts, like those from Defender for Identity, and automating alert management to streamline your security processes.

- **Filtering alerts for MDI** : To filter alerts specifically for MDI, use the **detectionSource** filter in your API requests. This is essential for focusing on relevant alerts and streamlining your response process. Here's how to apply this filter using PowerShell:

```
$uri = "https://graph.microsoft.com/beta/security/alerts_v2?
$filter=detectionSource eq 'DefenderForIdentity'"
$headers = @{
    "Authorization" = "Bearer <access_token>"
}
$response = Invoke-RestMethod -Uri $uri -Headers $headers -Method Get
$response
```

- **Automating alert management** : Automating alert management enhances your security operations by reducing manual intervention and ensuring timely responses. Here is an example of a PowerShell script that retrieves and processes MDI alerts:

```
$tenantId = "<your_tenant_id>"
$clientId = "<your_client_id>"
$clientSecret = "<your_client_secret>"
$body = @{
    grant_type      = "client_credentials"
    scope          = "https://graph.microsoft.com/.default"
    client_id      = $clientId
    client_secret   = $clientSecret
}
$response = Invoke-RestMethod -Method Post -Uri
"https://login.microsoftonline.com/$tenantId/oauth2/v2.0/token" -ContentType
"application/x-www-form-urlencoded" -Body $body
$accessToken = $response.access_token
# Retrieve Defender for Identity alerts
```

```

$headers = @{
    Authorization = "Bearer $accessToken"
}
$alerts = Invoke-RestMethod -Method Get -Uri
"https://graph.microsoft.com/beta/security/alerts_v2?$filter=detectionSource eq
'defenderForIdentity'" -Headers $headers
# Process alerts
foreach ($alert in $alerts.value) {
    Write-Host "Alert: $($alert.title) - $($alert.status)"
    # Add logic to handle or escalate alerts
}

```

By automating the alert management process, you can ensure that critical alerts are addressed promptly, enhancing your overall security posture.

Investigating incidents with the Incident API endpoint

In this section, you will explore the Incident API endpoint, accessible via `/security/incidents`. This API helps you investigate and manage security incidents that include alerts from MDI, providing a comprehensive view of security events. Let's dive into some practical examples of how you can use the Incident API to enhance your security operations:

- **Retrieving MDI-related incidents**: Use the `expand` parameter to retrieve incidents along with their related alerts. This is essential for understanding the context and impact of security incidents. This is how you can achieve this using PowerShell:

```

$uri = "https://graph.microsoft.com/beta/security/incidents?$expand=alerts"
$headers = @{
    "Authorization" = "Bearer <access_token>"
}
$response = Invoke-RestMethod -Uri $uri -Headers $headers -Method Get
$response

```

- **Incident response automation**: Automating incident response can significantly reduce response times and improve coordination during security events. Here is an example of a PowerShell script to for retrieving and processing MDI-related incidents:

```

$tenantId = "<your_tenant_id>"
$client Id = "<your_client_id>"
$clientSecret = "<your_client_secret>"
$body = @{
    grant_type      = "client_credentials"
    scope          = "https://graph.microsoft.com/.default"
    client_id      = $clientId
    client_secret   = $clientSecret
}
$response = Invoke-RestMethod -Method Post -Uri
"https://login.microsoftonline.com/$tenantId/oauth2/v2.0/token" -ContentType
"application/x-www-form-urlencoded" -Body $body
$accessToken = $response.access_token
# Retrieve incidents with expanded alerts
$headers = @{
    Authorization = "Bearer $accessToken"
}
$incidents = Invoke-RestMethod -Method Get -Uri
"https://graph.microsoft.com/beta/security/incidents?$expand=alerts" -Headers
$headers
# Process incidents
foreach ($incident in $incidents.value) {
    Write-Host "Incident: $($incident.id) - $($incident.status)"
    foreach ($alert in $incident.alerts) {
        if ($alert.detectionSource -eq "defenderForIdentity") {

```

```
        Write-Host "MDI Alert: $($alert.title)"
        # Add logic to handle or escalate alerts
    }
}
}
```

By automating incident response, you can ensure a coordinated and efficient response to security threats, minimizing their impact on your organization.

Having explored the capabilities of the Graph API, let's now turn our attention to utilizing this powerful tool in building custom integrations and automations for real-world implementations.

Building custom integrations and automations

In this section, we will explore how to use the Microsoft Graph API to build custom integrations and automations for MDI. By automating routine tasks and integrating MDI with other security tools, you can enhance your organization's ability to detect and respond to identity-based threats efficiently. This section will cover practical examples to help you implement these integrations and automations effectively.

Identifying integration opportunities

In this subsection, you will identify common scenarios where integrating MDI with other tools can enhance your security operations. Understanding these opportunities will help you design integrations that maximize the value of MDI within your security ecosystem.

Integrating MDI and the entire Defender XDR with other security tools allows a more comprehensive and unified security posture. By recognizing key integration points, you can streamline threat detection, response, and management processes.

Here are some example integrations:

- **SIEM integration** : Sending MDI alerts to a SIEM solution such as Splunk or Microsoft Sentinel for centralized monitoring and correlation. By default, we are sending alerts to Defender XDR.
- **IT Service Management (ITSM) integration** : Creating automated incident tickets in ITSM tools like ServiceNow based on MDI alerts.
- **Automating healthIssues response** : Sending MDI misconfigurations to either SIEM solution or ITSM.
- **Integration with custom dashboards and reporting tools** : Use the Graph API to extract detailed security insights and telemetry from, for example, Defender XDR, and display them on custom dashboards or Power BI reports.

Type of use cases

I want you to start thinking about the possibilities we get with data from Graph API, there could be endless automation and reporting, integration with other systems, and so on. By thinking creatively

about how to use data from the Graph API, you can unlock a wide range of possibilities for automating and integrating your security operations, ultimately leading to a more efficient and effective security posture. Since I'm a huge fan of PowerShell, Azure Functions, and Event Grid, let's take the opportunity to learn how to integrate Graph API with some PowerShell Scripting and other Azure services.

Here are some practical examples to help you start thinking .

Integrating MDI healthIssues with monitoring tools

This solution uses Logic Apps with a system-assigned managed identity, authorized via the `SecurityIdentitiesHealth.Read.All` Graph API permission, to efficiently monitor and manage MDI health issues. You can deploy this solution through the GitHub repository for this chapter by clicking the **Deploy to Azure** button. Here's how the integration works:

- Data collection and condition logic:
 - System-assigned managed identity enabled with the following permissions:
 - Storage Table Data Contributor (on resource group)
 - Monitoring Metrics Publisher (on resource group)
 - SecurityIdentitiesHealth.Read.All (Graph API)
 - The Logic App uses an HTTP action connector to fetch MDI health issues from the Graph API
 - It then checks Azure Table for an existing entry matching the issue ID
 - Using conditional logic, the Logic App determines whether the issue is new (requiring creation) or has changed status (requiring an update)
- Data ingestion and storage:
 - If the issue is new or updated, the Logic App sends the JSON payload to a Log Analytics workspace. This is done via another HTTP action that connects to the **Data Collection Endpoint (DCE)**, referencing a predefined **Data Collection Rule (DCR)** that dictates the schema for the **MDIHealthIssues_CL** custom table.
 - Subsequently, the issue details are either updated or newly created in Azure Table Storage using the Insert Or Replace Entity connector.
- Visualization and alerting:
 - Administrators can set up alerts in Azure Monitor to notify them when new health issues are detected
 - Azure Workbooks can be utilized to visualize the data, providing a clear and interactive display of health issue trends and specifics

The following image shows you an example flow in Logic Apps for the described solution (see *Figure 5 .6*):

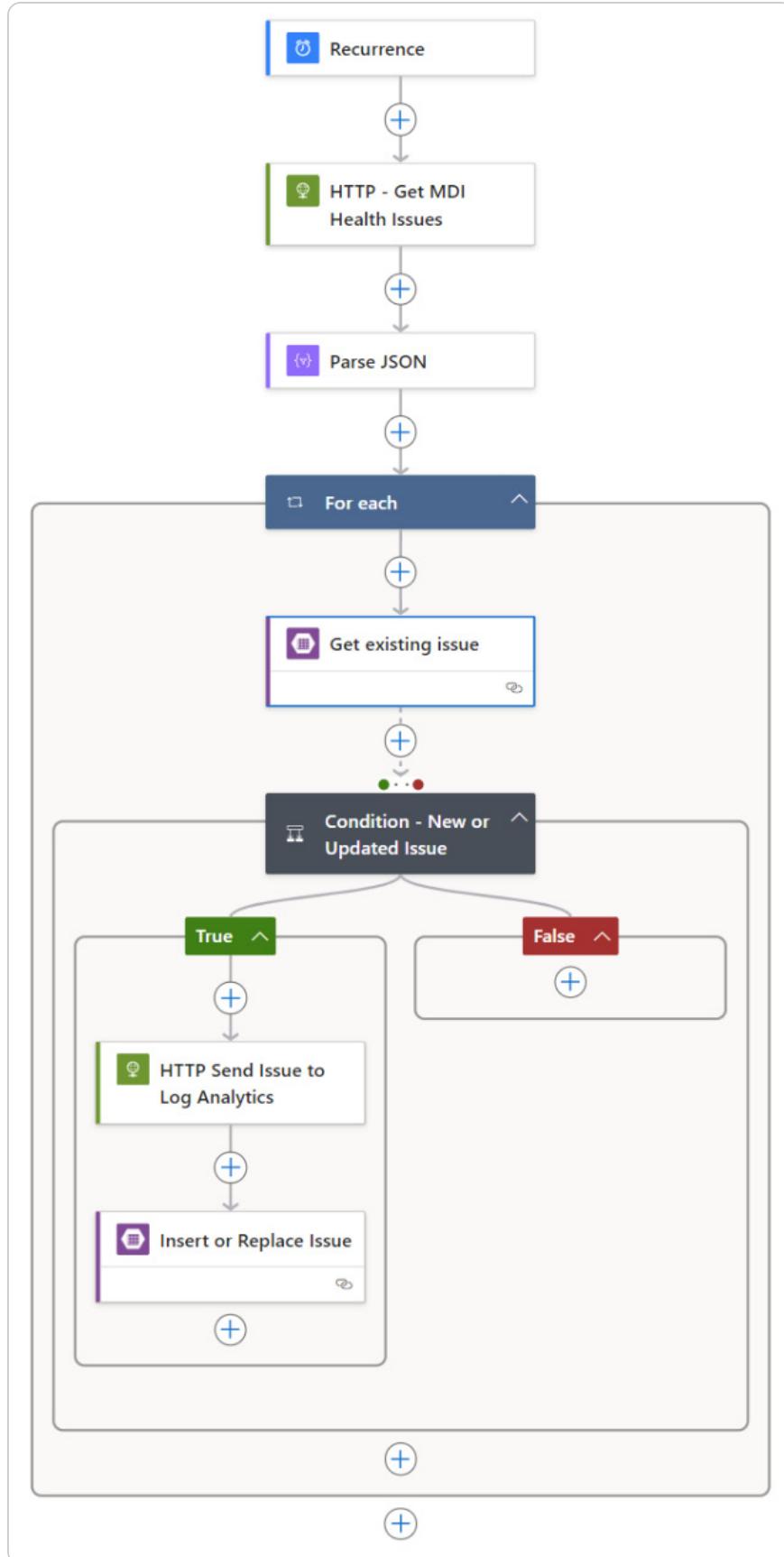


Figure 5.6 – Logic App Designer view

Next, let's explore another use case that enables us to automate the response to health issues.

Automating health issues responses

This solution uses a PowerShell script that automatically resolves minor health issues or escalates critical issues to the appropriate team. This automation helps ensure timely and consistent responses to health issues, reducing the burden on your IT staff and improving your overall security posture (please refer to the ServiceNow documentation for the correct API call):

```
$tenantId = "<your_tenant_id>"  
$clientId = "<your_client_id>"  
$clientSecret = "<your_client_secret>"  
$body = @{  
    grant_type      = "client_credentials"  
    scope          = "https://graph.microsoft.com/.default"  
    client_id      = $clientId  
    client_secret   = $clientSecret  
}  
$response = Invoke-RestMethod -Method Post -Uri  
"https://login.microsoftonline.com/$tenantId/oauth2/v2.0/token" -ContentType  
"application/x-www-form-urlencoded" -Body $body  
$accessToken = $response.access_token  
# Retrieve health issues  
$headers = @{  
    Authorization = "Bearer $accessToken"  
}  
$healthIssues = Invoke-RestMethod -Method Get -Uri  
"https://graph.microsoft.com/beta/security/identities/healthIssues" -Headers $headers  
# Process health issues  
foreach ($issue in $healthIssues.value) {  
    if ($issue.severity -eq "low") {  
        # Automatically resolve minor issues  
        $updatePayload = @{  
            status = "resolved"  
            comments = "Automatically resolved as minor issue"  
        } | ConvertTo-Json  
        Invoke-RestMethod -Method Patch -Uri  
"https://graph.microsoft.com/beta/security/identities/healthIssues/$($issue.id)" -  
Headers $headers -Body $updatePayload  
    } elseif ($issue.severity -eq "high") {  
        # Escalate critical issues  
        $incidentPayload = @{  
            short_description = $issue.issueType  
            description = $issue.additionalDetails  
            severity = "1"  
            category = "Security"  
        } | ConvertTo-Json  
        $serviceNowUrl = "<your_servicenow_instance>/api/now/table/incident"  
        $serviceNowUser = "<your_servicenow_username>"  
        $serviceNowPassword = "<your_servicenow_password>"  
        Invoke-RestMethod -Method Post -Uri $serviceNowUrl -Headers @{"Content-Type"  
= "application/json"} -Credential (New-Object  
System.Management.Automation.PSCredential($serviceNowUser, (ConvertTo-SecureString  
$serviceNowPassword -AsPlainText -Force))) -Body $incidentPayload  
    }  
}
```

We have now explored various integration opportunities and use cases for utilizing the Microsoft Graph API with the current API available to MDI. By connecting MDI data from Graph API to other tools, we can create innovative solutions, such as automating routine tasks, enhancing our ability to manage and respond to security challenges efficiently.

Summary

In this chapter, we've taken an in-depth look at the capabilities of the MDI API, focusing on the new `healthIssue` API, which is currently in beta. We discussed how this API can be used to identify and address potential issues with the MDI configuration, enhancing security operations.

Additionally, we explored the essential steps for setting up an app registration in Microsoft Entra ID. This setup is crucial for granting the API permissions that enable us to interact effectively with the MDI through the Microsoft Graph API. By obtaining these permissions, we pave the way for developing custom solutions tailored to support IT and security operations. These solutions can significantly streamline the management of the MDI environment, improving both efficiency and response capabilities. You also learned the difference between app registrations and enterprise applications.

We also delved into practical implementations, demonstrating how the Graph API can be used to integrate MDI data with other tools and systems. This integration opens up opportunities for automating routine tasks and enhancing security measures, which are critical for modern IT and security operations.

In the next chapter, we will start hunting and get to know our MDI data through **Kusto Query Language (KQL)**.

HappyHunting

6

Mastering KQL for Advanced Threat Detection in MDI

In this chapter, we will delve into the capabilities of **Kusto Query Language (KQL)** within MDI. Our journey will start with an introduction to KQL where you will learn how to write simple queries to retrieve and filter data from MDI tables. This section is crucial for establishing a solid foundation, enabling you to leverage KQL's capabilities in your security operations.

As we advance in the chapter, you will discover how to utilize KQL for more sophisticated threat detection. The middle part of the chapter will equip you with the skills to identify hidden patterns, anomalies, and correlations in your data, essential for uncovering advanced threats. You will master techniques such as joining multiple tables, summarizing data, and creating custom columns.

To bring theory into practice, we will explore real-world case studies where KQL and MDI together with **Microsoft Defender for Endpoint (MDE)** have been pivotal in detecting advanced attacks. These examples will provide practical insights into applying the techniques you've learned to actual security challenges.

As we advance, you will discover how to utilize KQL for more sophisticated threat detection. This part of the chapter will equip you with the skills to identify hidden patterns, anomalies, and correlations in your data, essential for uncovering advanced threats. You will master techniques such as joining multiple tables, summarizing data, and creating custom columns. These advanced skills will enhance your ability to detect and respond to security incidents efficiently.

By the end of this chapter, you will have a comprehensive understanding of how to use KQL to query and analyze MDI data. You will be equipped to perform advanced threat detection and apply your skills to real-world scenarios. These capabilities are essential for any cybersecurity professional looking to enhance their threat-hunting and incident response efforts. Mastering KQL will empower you to make data-driven decisions, improve your organization's security defenses, and stay ahead of potential threats.

In this chapter, we're going to cover the following main topics:

- KQL for beginners – querying MDI data
- Advanced KQL techniques for deep threat detection
- Real-world case studies – detecting advanced attacks with KQL

Let's get started!

Technical requirements

For this chapter, you need to have everything deployed that we went through during the initial chapters; as a reminder, you will need to have the following:

- Microsoft tenant
- Microsoft subscription that includes Microsoft Defender for Identity, such as Microsoft 365 E5 or Microsoft 365 E3 + E5 Security
- Basic Microsoft 365 knowledge
- Active Directory knowledge
- A virtual or physical server environment with Active Directory installed and configured
- Basic PowerShell knowledge:
 - Windows PowerShell 5.1 or PowerShell 7.4 or later

All the code examples for this chapter can be found on GitHub at

<https://github.com/PacktPublishing/Microsoft-Defender-for-Identity-in-Depth/tree/main/Chapter06>

KQL for beginners – querying MDI data

MDI is a powerful tool that provides deep insights into your organization's security posture. However, to fully leverage its capabilities, you need to know how to query and analyze the data it provides. This is where KQL comes into play. In this section, we'll start with the basics of KQL and guide you through the initial steps of querying MDI data within the broader Microsoft Defender XDR environment. But before diving into the different tables and queries, let's take a history lesson about KQL, or Kusto as it's also known.

The history of KQL and its ecosystem

KQL has become a cornerstone for data analysis and cybersecurity within the Microsoft ecosystem. Its evolution is closely linked to the development of the underlying technology, codenamed *Kusto*, which is the foundation of **Azure Data Explorer (ADX)**. Understanding the history and significance of KQL provides valuable insights into its capabilities and its fundamental role in modern data analytics.

Origins of Kusto and KQL

The journey of KQL began with the creation of Kusto, a codename used during the development of this powerful big data analytics service at Microsoft. Named after Jacques-Yves Cousteau, the renowned oceanographer, the project aimed to explore immense amounts of data in a manner like Cousteau's Ocean explorations. Kusto was designed to provide a platform for real-time analytics capable of handling large-scale data from various sources such as applications, websites, and IoT devices.

Development and evolution of KQL

The goal was to make data querying both powerful and user-friendly. KQL's syntax and structure draw inspiration from SQL, making it accessible to those familiar with traditional database query languages. However, it also incorporates unique features tailored for large-scale data exploration, such as time-series analysis and advanced data transformation capabilities.

KQL quickly gained traction within Microsoft and among external developers for its flexibility and efficiency in querying large datasets. Products such as Azure Monitor, Azure Cost Management, Microsoft Defender, LinkedIn, Microsoft Sentinel, and so on use Kusto.

ADX

ADX is the platform that brought KQL to a broader audience. Launched in public preview in 2018 and made generally available in 2019, ADX provides a fast and highly scalable service for data ingestion and query processing. It is designed to handle massive volumes of diverse data types, making it ideal for scenarios requiring real-time analytics.

ADX's architecture supports high-speed data ingestion and efficient query processing, allowing users to derive insights from their data almost instantaneously. This capability has positioned ADX as a critical tool for modern data-driven applications and services.

ADX is a distributed database, based on relational database management systems, but it doesn't have constraints such as mandatory primary and foreign keys, allowing more flexibility for data ingestion and analysis without needing to predefine strict relationships between tables.

In a traditional relational database, you often have to define primary keys to uniquely identify each record in a table. You also need to set up foreign keys to maintain relationships between tables. Here's a simple example using a customers and orders table:

CustomerID (Primary Key)	Name	Email
1	Alice	alice@example.com
2	Bob	bob@example.com

Table 6.1 – Customers table

In this example, `customerID` is a primary key in the `customers` table and serves as a foreign key in the related `orders` table, ensuring that each order is linked to a specific customer:

OrderID (Primary Key)	CustomerID (Foreign Key)	Product	Quantity

101	1	Laptop	1
102	2	Phone	2

Table 6.2 – Orders table

Here are some key terms mentioned in the preceding tables:

- **Primary key** : A unique identifier for each record in the table – for example, `CustomerID` and `OrderID` .
- **Foreign key** : A field that creates a relationship between two tables. For example, `CustomerID` in the `Orders` table links to `CustomerID` in the `Customers` table.

In ADX, there's no need to define primary or foreign keys in advance. Instead, you can store your data freely and focus on querying and analyzing it when necessary. For example, here's how the `Customers` table looks in ADX:

CustomerID	Name	Email
1	Alice	<code>alice@example.com</code>
2	Bob	<code>bob@example.com</code>

Table 6.3 – Customers table in ADX

In contrast to traditional relational databases, ADX allows you to store data without predefined relationships. You can still link and analyze the data dynamically using queries, as demonstrated by the `orders` table here:

OrderID	CustomerID	Product	Quantity
101	1	Laptop	1
102	2	Phone	2

Table 6.4 – Orders table in ADX

With ADX, you focus on querying the data to uncover relationships and insights when needed, offering flexibility for managing large-scale and evolving datasets.

How queries might look

In a traditional **relational database management system** (**RDBMS**), queries are structured to work within a rigid schema that defines table relationships, data types, and indexing. This structure often uses **Structured Query Language** (**SQL**) for managing and manipulating the data.

Let's look at an RDBMS query example. To find all orders for a customer in a traditional RDBMS, you might write the following:

```
SELECT Orders.OrderID, Orders.Product, Orders.Quantity
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID
WHERE Customers.Name = 'Alice';
```

This query demonstrates how SQL leverages `JOIN` to combine rows from two or more tables, based on a related column between them. Here, the `orders` and `customers` tables are linked through `CustomerID` to filter out orders specifically for `Alice`.

In contrast, ADX, which is optimized for large-scale data analytics, provides a more flexible environment that doesn't enforce a strict schema. This makes it particularly useful for working with big data scenarios where schema can evolve or might not be fully normalized. ADX uses KQL, which is designed to handle large volumes of unstructured data efficiently.

Let's now look at a KQL query. To find the same information in ADX or Log Analytics with KQL, you might write the following:

```
Customers
| join kind=inner (Orders) on CustomerID
| where Name == "Alice"
```

In this KQL example, the query uses an `inner` join to filter and display records where the customer's name is `Alice`, much like the SQL example but with a syntax that's optimized for quick reads and less formal structure.

In summary, ADX simplifies data management by removing the need for rigid structures such as primary and foreign keys, focusing instead on flexible data storage and powerful querying capabilities. This makes it easier, faster, and enjoyable to work with large amounts of data.

The pattern of getting data into an ADX cluster and database is quite easy: creating the database, ingesting data, and querying the database.

Broader applications and ecosystem

KQL's versatility has led to its integration across various Microsoft products and services, particularly in cybersecurity. MDI and other components of the Microsoft Defender suite use KQL for advanced threat hunting and security analytics. By providing a consistent query language across different platforms, Microsoft has enabled a unified approach to data analysis and threat detection.

Beyond cybersecurity, KQL is used in application performance monitoring, business intelligence, and operational analytics. Its ability to efficiently query and analyze diverse data types has made it a valuable tool for organizations aiming to leverage big data for strategic decision-making.

Kusto Detective Agency

To make learning KQL more engaging and accessible, Microsoft introduced the **Kusto Detective Agency (KDA)**, which is an innovative and interactive platform created by Microsoft to teach users how to use KQL in a fun and engaging way. By solving detective cases through data analysis, users can develop their KQL skills in a practical, hands-on manner. This makes KDA an excellent resource for both beginners and experienced users looking to enhance their data querying capabilities.

The KDA presents users with a series of detective cases that they must solve using KQL. Each case involves analyzing datasets to uncover clues and solve mysteries. This gamified approach makes learning KQL enjoyable and practical, providing users with real-world scenarios that mirror common tasks in data analysis.

Key features of KDA

To better understand what makes KDA so effective and enjoyable for learners, here are some key features that highlight its unique educational approach:

- **Gamified learning experience :** KDA transforms learning into an exciting game. Users earn badges and rewards as they solve cases, which motivates them to continue improving their skills.
- **Seasonal challenges :** KDA operates in seasons, with each season introducing new cases and challenges. For example, Season 2 offered more complex cases.
- **Beginner-friendly :** The platform is designed to be accessible to those new to KQL. It starts with simpler queries and gradually increases in complexity, making it suitable for users at all skill levels.
- **Community and collaboration :** Participants often share their solutions and their progress within the community (via LinkedIn and blog posts), fostering collaboration and peer learning.

The skills developed through KDA are directly transferable to scenarios involving Azure Log Analytics, Defender XDR, Application Insights, and ADX, which uses KQL for querying and analyzing data.

To join the KDA, you can start by creating a KQL database in Synapse Real-time Analytics in Microsoft Fabric or by setting up a free Kusto cluster. This setup provides the environment needed to dive into the detective cases and start improving your KQL skills. Visit <https://detective.kusto.io> to begin your journey.

Understanding your MDI data

Before diving into KQL, it's essential to understand the types of data available in MDI. MDI collects a wealth of information related to user activities, network interactions, and security events. This data is crucial for identifying potential threats and understanding the behavior of users and devices in your network. When you master the skills of KQL, you can then join tables to correlate events.

The key data types include the following:

- **Alerts** : Notifications about potentially suspicious activities or policy violations
- **Events** : Logs of user and device activities, such as logins, file accesses, and network connections
- **Entities** : Information about users, devices, and resources within your network

Key concepts of a table

To fully leverage the power of KQL in analyzing MDI data, it's important to understand the fundamental structure of tables where this data is stored. Here are the basic components of a table that play a critical role in organizing and querying data effectively:

- **Columns** : Think of columns as categories or types of information stored in the table. Each column has a name and contains a specific kind of data. For example, in a table recording user logins, you might have columns for the username, login time, and login status.
- **Rows** : Each row represents a single record or entry in the table. Using the login example, each row would represent one login attempt by a user.
- **Schema** : The schema is the structure of the table, defining what columns exist and what type of data each column holds (e.g., text, numbers, and dates).

Imagine you have a table named `IdentityLogonEvents` that logs user login activities. Here's how it might look:

TimeGenerated	UserName	ActionType	Result
2024-06-04 12:00:00	alice	Failed	Success
2024-06-04 12:05:00	bob	Logon	Failure

Table 6.5 – Example of the IdentityLogonEvents table

This example table (see *Table 6.5*) illustrates how individual data points are organized within the MDI dataset. By examining the structure of this table, you can gain insights into how data is formatted and how each column serves a specific function in the broader context of data analysis. This understanding is essential for formulating effective queries that can accurately identify and interpret significant events within your environment. Let's take a closer look at the core components of the table:

- **Columns**: `TimeGenerated` , `UserName` , `ActionType` , and `Result`
- **Rows** : Each row represents one login attempt with details such as the time it happened, the user involved, the type of action, and the result of the action

Knowing what data is available helps you frame the right questions and craft effective queries to be able to find eventual evidence regarding threats in your environment.

Getting started with KQL

KQL is designed to be intuitive and easy to learn, even for those new to query languages. At its core, KQL is all about making simple, readable queries that can quickly return relevant data. Let's start with a basic example to get you comfortable with the syntax and structure of KQL.

Consider you want to retrieve all security alerts generated in the last 24 hours. A simple KQL query for this would look like the following:

SecurityAlert

```
| where Timestamp >= ago(24h)
```

This query does two things: it selects data from the `SecurityAlert` table and filters it to only include alerts generated in the last 24 hours. The `|` symbol is a pipe, used to sequence operators, passing the output of one operator as the input to the next. Here's a breakdown of the query:

- `SecurityAlert` : This specifies the table from which to retrieve the data.
- `|` (pipe): This symbol is used to chain operations together. Each operation manipulates the data in some way before passing it to the next operation.
- `where` : This operator filters the data based on a condition.
- `Timestamp >= ago(24h)` : This condition filters the data to include only rows where the `Timestamp` column value is greater than 24 hours ago. The `ago(24h)` function returns the current time minus 24 hours.

By using this query, you are effectively asking for all rows in the `SecurityAlert` table where the `Timestamp` value is within the last 24 hours.

Let's create another basic query example from the `IdentityLogonEvents` table. Suppose you want to retrieve all logon events that were successful in the last 24 hours. Here's how you can write that query:

```
IdentityLogonEvents
| where Timestamp >= ago(24h)
| where ActionType == "LogonSuccess"
```

This query selects data from the `IdentityLogonEvents` table and filters it to include only successful logon events from the last 24 hours. Here's the breakdown of the query:

- `IdentityLogonEvents` : This is the table containing logon event data
- `Timestamp` : The timestamp column indicating when each logon event occurred
- `ActionType` : The column indicating the outcome of the logon attempt (e.g., `LogonSuccess` or `LogonFailed`)
- `ago(24h)` : A datetime/timespan scalar function that returns the current time minus 24 hours

By using this query, you can monitor recent successful logon attempts and potentially correlate them with other security data (tables) to identify unusual patterns or activities.

To view the full schema of tables, you can use the built-in **Schema reference** within **Advanced hunting** in Defender XDR. This submenu allows you to search for specific tables, such as the `IdentityLogonEvents` table. You can then examine columns such as `ActionType` to see the possible values it can contain. To access the **Schema reference** menu, follow these steps:

1. Log in with appropriate permissions to security.microsoft.com.
2. Go to **Investigation & response** > **Hunting** > **Advanced hunting**.
3. If you see the **Schema reference** icon, click on it. If you don't see the icon, click on the ellipsis (three dots) at the top right of the page, then select the **Schema reference** icon from the drop-down menu (see *Figure 6.1*).

The screenshot shows the Microsoft Defender XDR Advanced hunting interface. At the top, there are several tabs labeled 'New query' (with one highlighted in blue), 'Run query' (which is bolded), 'Last 7 days', 'Save', 'Share link', 'Help resources', 'Query resources report', and 'Schema reference' (which is enclosed in a red box). Below the tabs, there's a section titled 'Query' with the following code:
1 IdentityLogonEvents
2 | where ActionType == "LogonSuccess"
Underneath the code, there are three tabs: 'Getting started' (disabled), 'Results' (selected and underlined in blue), and 'Query history'. To the right of these tabs are buttons for 'Export', 'Show empty columns', 'Search' (with a placeholder 'Search'), and 'Chart type'. Below the tabs, there's a 'Filters' section with a 'Add filter' button and a table of filters:

Filter Type	Value 1	Value 2	Value 3	Value 4	Value 5	Value 6
<input type="checkbox"/> TimeGenerated	Oct 18, 2024 10:08...	Oct 18, 2024 10:08:56 AM	LogonSuccess	Active Directory	Resource access	Kerberos

Figure 6.1 – Defender XDR Schema reference

Search for the table you want to know more about in the search field (1) and then click on the table in the list (2) (see *Figure 6.2*).

Schema reference

1

Identity

X

AADManagedIdentitySignInLogs

IdentityDirectoryEvents

IdentityInfo

2

IdentityLogonEvents

IdentityQueryEvents

Figure 6.2 – Schema reference

See the different `ActionType` values for the table you searched for (see *Figure 6.3*).



IdentityLogonEvents

Advanced table ⓘ

▶ See preview data

ActionType values



LogonFailed

A user attempted to logon to the device but failed.

LogonSuccess

A user successfully logged on to the device.

Sample queries



Cleartext passwords in LDAP authentication

Find LDAP authentication attempts using cleartext passwords



See full documentation

Figure 6.3 – IdentityLogonEvents in Schema reference

Now that we have a grasp on how the schema is structured, it's time to dive deeper and start exploring some data.

Exploring data with KQL

Once you're comfortable with basic queries, you can start exploring your data more deeply. One thing I personally tend to do is always look at the table and grab the first rows to see what type of columns and information the table consumes. To do this, I'm using the `take` operator; see the following query:

```
IdentityLogonEvents  
| take 5
```

With this query, we will get the *5 latest events* from the `IdentityLogonEvents` table with all the columns printed to the output.

Now, if we want to focus on some of the columns because the other columns are not that important to us for the moment, we can choose a few of them and print them to our output. To do that, we will use the `project` operator and define which of the columns we want to see:

```
IdentityLogonEvents  
| project Timestamp, ActionType, Application, DestinationPort  
| take 5
```

With that query, we will have the output and value from the `Timestamp`, `ActionType`, `Application`, and `DestinationPort` columns, which are a part of the `IdentityLogonEvents` table.

Defining new columns

In KQL, you can define new columns using the `extend` operator. This operator allows you to create new columns based on existing ones or by performing calculations and transformations:

```
| extend NewColumnName = Expression
```

Suppose you have an `IdentityLogonEvents` table and you want to create a new column that indicates the duration of each logon session in minutes:

```
IdentityLogonEvents  
| extend SessionDurationMinutes = (LogoffTime - LogonTime) / 1m
```

Renaming columns

To rename existing columns, KQL provides the `project-rename` operator. This operator allows you to change the names of one or more columns, which can help in making the data output more readable or in aligning it with specific naming conventions:

```
| project-rename NewColumnName = ExistingColumnName
```

Consider the `SecurityEvent` table where you want to rename the `Timestamp` column to `EventTime` and the `Computer` column to `MachineName` :

```
SecurityEvent
| project-rename EventTime = Timestamp, MachineName = Computer
```

Filtering and sorting data

Filtering is a fundamental aspect of querying data. You can add multiple conditions to refine your search. For example, to find all high-severity alerts involving a specific user, your query might look like this:

```
SecurityAlert
| where Timestamp >= ago(24h)
| where Severity == "High"
| where User == "JohnDoe"
```

Sorting the results can also be helpful. To sort these high-severity alerts by their generation time, you can use the `sort` operator:

```
SecurityAlert
| where Timestamp >= ago(24h)
| where Severity == "High"
| where User == "JohnDoe"
| sort by Timestamp desc
```

Combining multiple tables

One of the strengths of KQL is its ability to combine data from multiple tables based on a matching column, providing a more comprehensive view. The general syntax is as follows:

```
Table1
| join kind=JoinType (Table2) on KeyColumn
```

Before we dive into the specifics of each component involved in a `join` operation, let's clarify the roles and terminology used when combining multiple tables in KQL. This will help you understand how data is merged from different sources to form a unified dataset:

- `Table1` : The primary table (will be referenced as the left table)
- `Table2` : The secondary table to be joined with the primary table (will be referenced as the right table)
- `JoinType` : The type of join (e.g., inner, outer, left outer, or right outer)
- `KeyColumn` : The column used as the key to join the tables

Explanation of left and right tables in KQL joins

In KQL, when performing a `join` operation, you reference the primary table first and then the secondary table. The terms “left table” and “right table” refer to the positions of these tables in the `join` operation:

- **Left table** : This is the table that is referenced first in the query. It is the primary table from which you want to start your data retrieval.
- **Right table** : This is the table that is referenced second in the `join` operation. It is the secondary table that you want to join with the primary table.

If we take one of the examples that we will dive into later in the chapter, we will have a query that looks like this:

```
IdentityLogonEvents
| where TimeGenerated >= ago(24h)
| join kind=inner (
    SecurityAlert
    | where TimeGenerated >= ago(24h)
    | where Severity == "High"
) on $left.User == $right.User
| project $left.TimeGenerated, $left.User, $left.LogonType, $right.AlertName,
$right.Severity, $right.Description
```

In this query, we have the following:

- **Primary table (left table)** : `IdentityLogonEvents` is the primary table because it is listed first
- **Secondary table (right table)** : `SecurityAlert` is the secondary table because it is listed second within the `join` operation

The terms `$left` and `$right` in the `on` clause refer to the columns in the primary and secondary tables, respectively.

TIP

To achieve optimal query performance, if one table is consistently smaller than the other, place the smaller table on the left side (primary, first table in the query) of the `join` operator.

Explanation of join types

Understanding the different **join types** is essential for building effective KQL queries, especially for threat hunting and creating use case alerts. Each join type serves a distinct purpose and affects the performance and results of your query in unique ways.

When joining tables, you combine rows from two or more tables based on a related column. The join type determines how these rows are matched and included in the query results. This understanding is crucial for several reasons.

First, join operations enable the correlation of data from different sources, providing a comprehensive view of your security landscape. For instance, combining user logon events with security alerts can highlight suspicious activities linked to specific logons.

Second, the choice of join type significantly impacts query performance. Using the appropriate join can enhance efficiency, reducing processing time and resource consumption. For instance, if one table

is smaller than the other, placing the smaller table on the left side of the join can lead to faster query execution.

Third, the accuracy of your results depends on the correct use of join types. Each type of join includes or excludes data based on the presence of matching rows in the tables. Understanding these distinctions ensures that your query results are relevant and precise.

Moreover, efficient joins help manage computational resources, particularly important in environments with substantial data volumes. Well-constructed joins prevent excessive memory and CPU usage, ensuring smoother operations.

Lastly, mastering join types provides the flexibility to construct complex queries. This flexibility is invaluable when building advanced use case alerts that require precise conditions to trigger correctly.

When constructing joins, consider the following:

- **Table size** : For optimal performance, if one table is smaller, use it as the left table in the join. This leverages the smaller dataset to filter the larger one more efficiently.
- **Data distribution** : Understand how data is distributed across your tables. Uneven distribution might require different join strategies to ensure optimal performance.
- **Query optimization** : Regularly review and optimize your queries to adapt to changing data patterns and volumes.

By understanding and using the different join types effectively, you can build powerful and efficient KQL queries that enhance your ability to hunt threats and create effective use case alerts. This foundational knowledge is key to leveraging KQL's full potential in security operations.

You will now see all of the join types described so you can make your choice when building the queries:

- **Inner join** :

An `inner` join returns only the rows where there is a match in both tables.

Here is an example:

```
Table1
| join kind=inner (Table2) on KeyColumn
```

A typical use case would be to find entries that exist in both `Table1` and `Table2` .

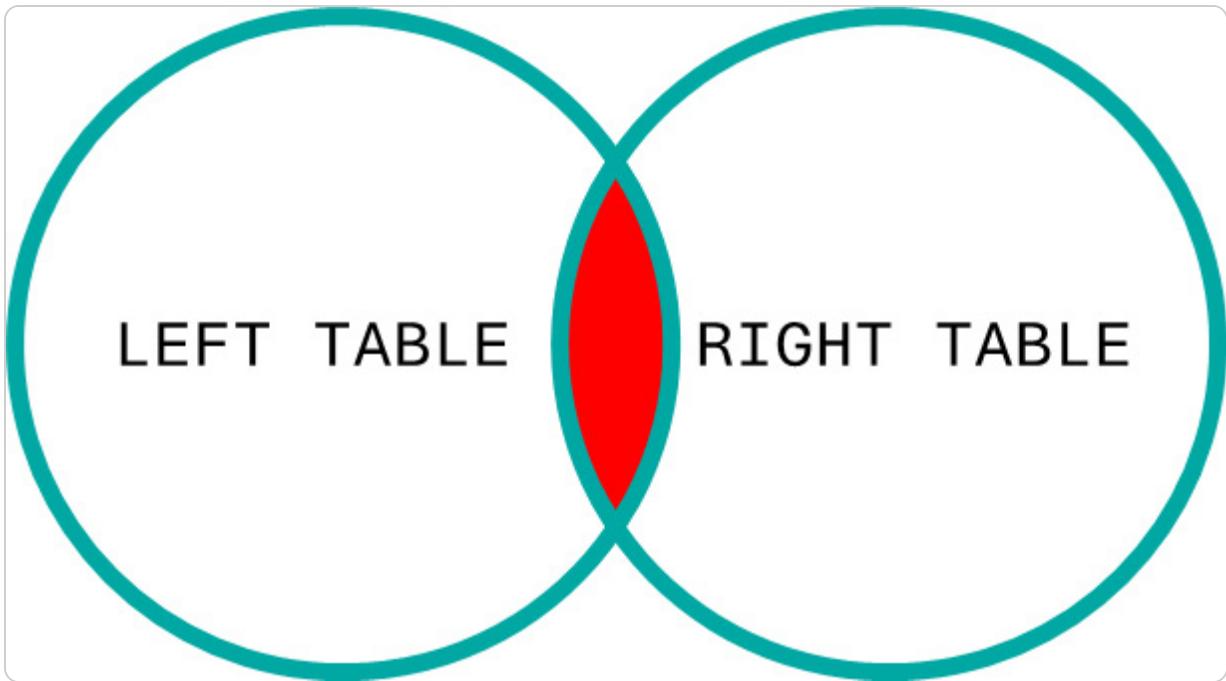


Figure 6.4 – Inner join

- **Left outer join :**

A `leftouter` join returns all rows from the left table and the matched rows from the right table. If there is no match, the result is still included but with null values for columns from the right table.

Here is an example:

```
Table1
| join kind=leftouter (Table2) on KeyColumn
```

A typical use case would be to find all entries from `Table1` and include corresponding data from `Table2`, if available.

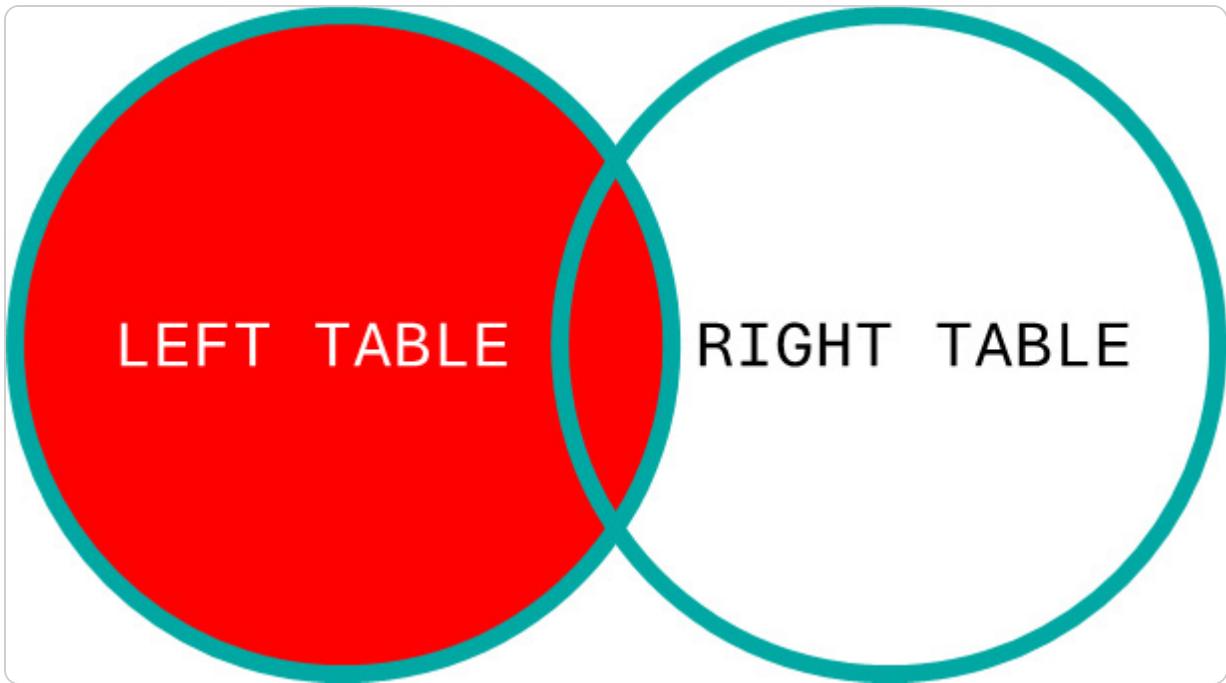


Figure 6.5 – Left outer join

- **Right outer join :**

A `rightouter` join returns all rows from the right table and the matched rows from the left table. If there is no match, the result is still included but with null values for columns from the left table.

Here is an example:

```
Table1
| join kind=rightouter (Table2) on KeyColumn
```

A typical use case would be to find all entries from `Table2` and include corresponding data from `Table1`, if available.

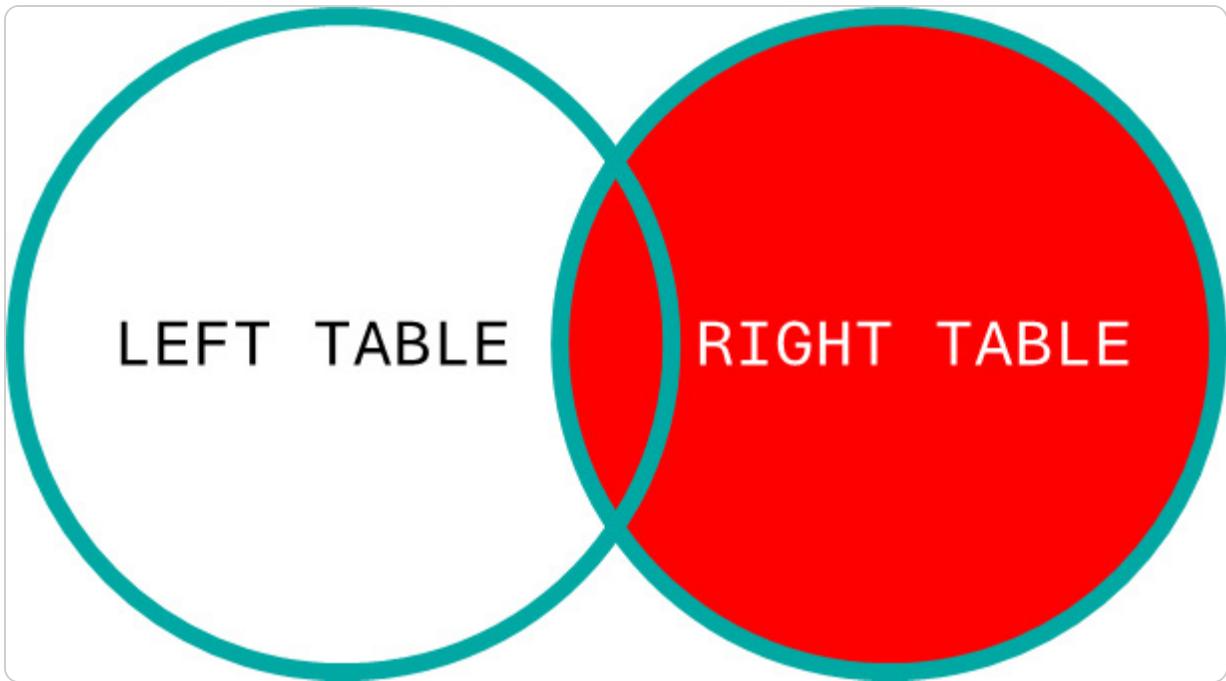


Figure 6.6 – Right outer join

- **Full outer join :**

A `fullouter` join returns all rows when there is a match in one of the tables. Rows in `Table1` that do not have matching rows in `Table2`, and vice versa, will be included in the result set with nulls in the columns where there is no match.

Here is an example:

```
Table1  
| join kind=fullouter (Table2) on KeyColumn
```

A typical use case would be to get a complete dataset including all matches and non-matches from both tables.

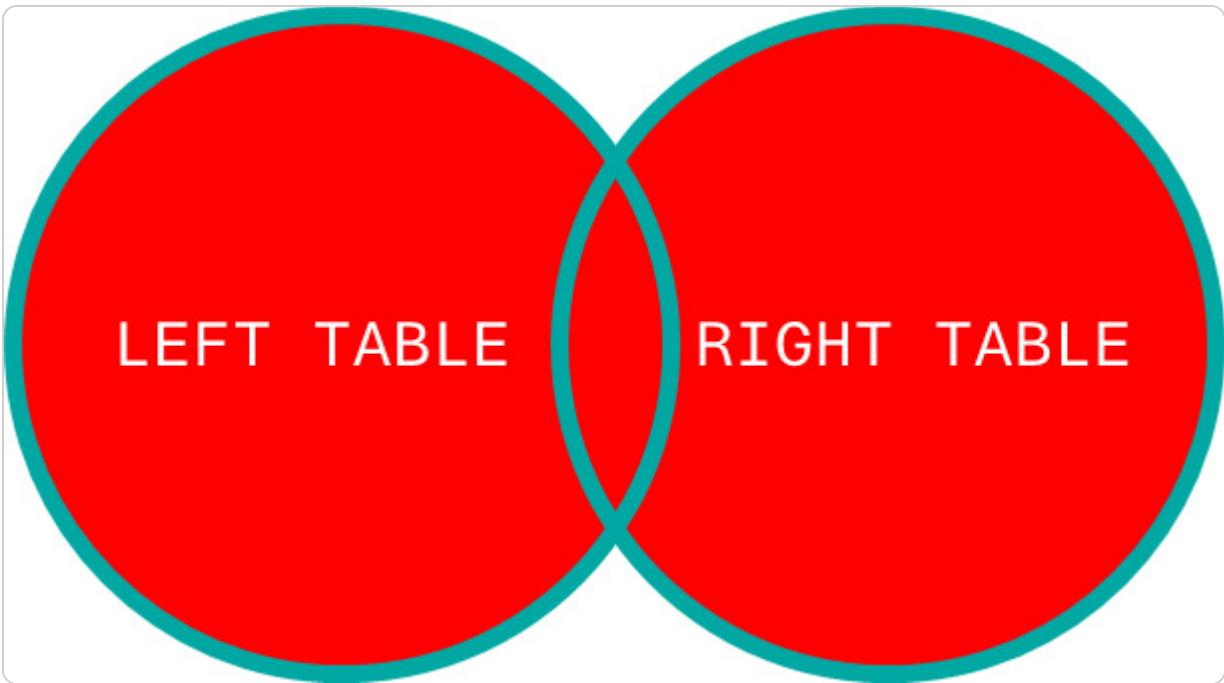


Figure 6.7 – Full outer join

- **Inner unique join (default join mode) :**

An `innerunique` join returns a unique subset of rows from the left table that have matching rows in the right table. This is particularly useful when you need to remove duplicates from the left table.

Here is an example:

```
Table1  
| join kind=innerunique (Table2) on KeyColumn
```

A typical use case would be to find unique matches in `Table1` that exist in `Table2` .

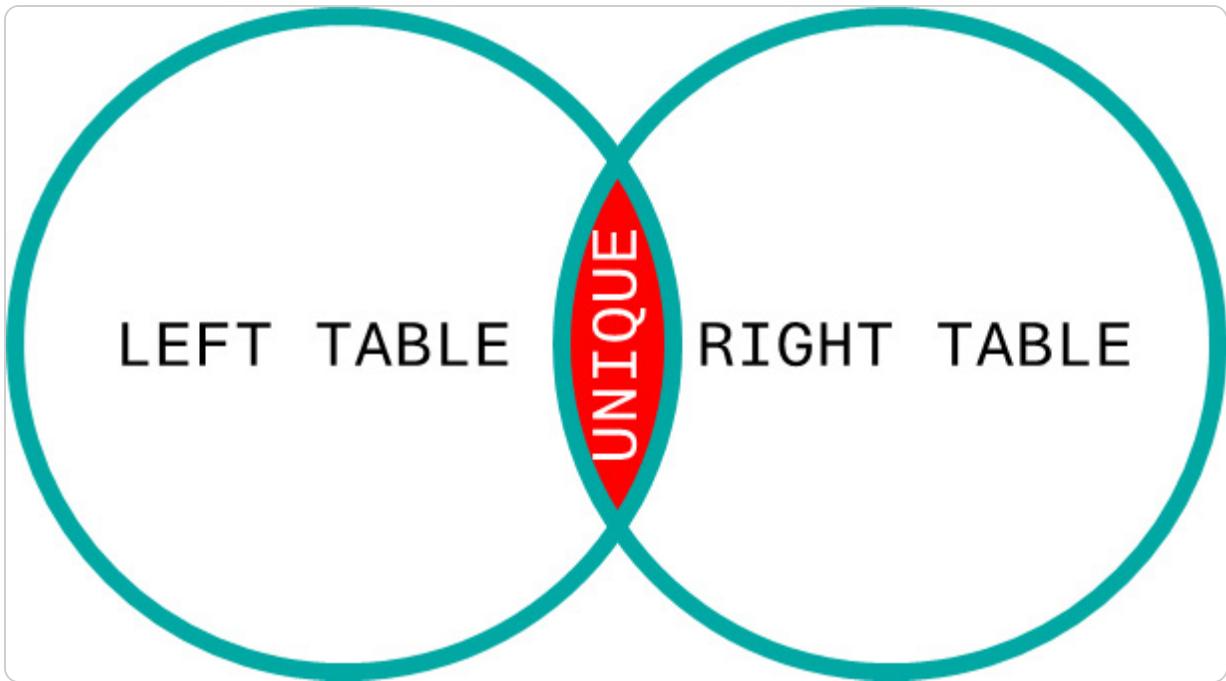


Figure 6.8 – Inner unique join

- **Left anti join :**

A `leftanti` join returns only the rows from the left table that do not have a match in the right table.

Here is an example:

```
Table1
| join kind=leftanti (Table2) on KeyColumn
```

A typical use case would be to find entries in `Table1` that do not have corresponding entries in `Table2`.

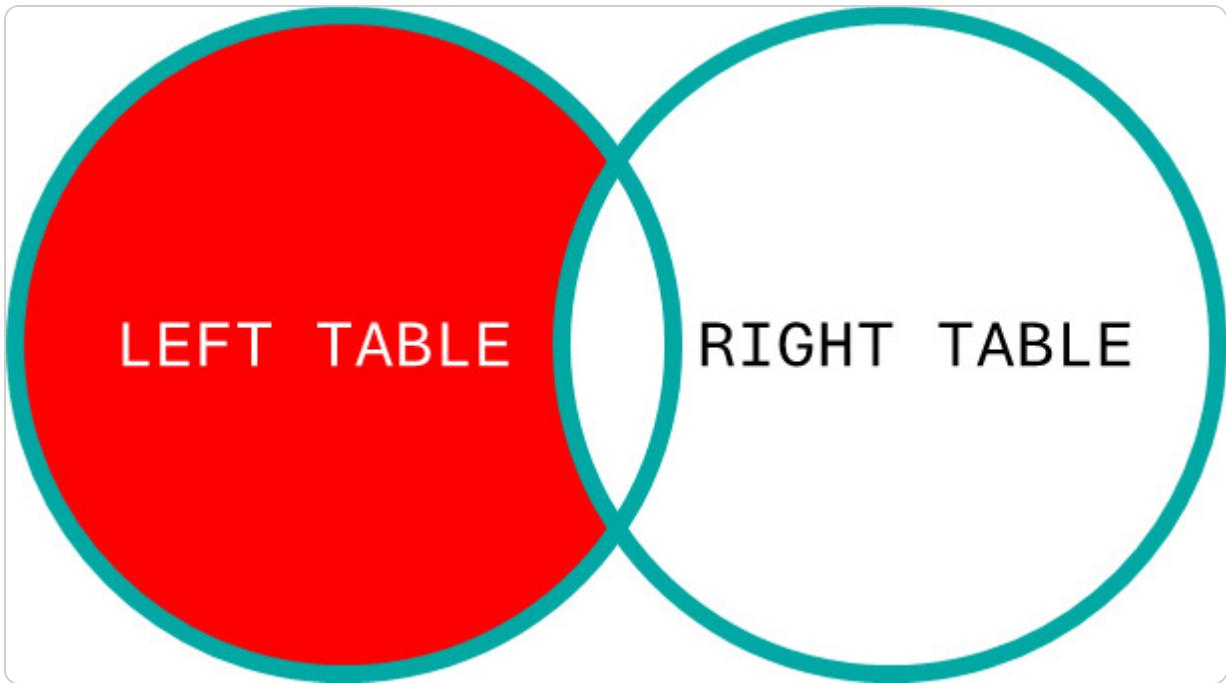


Figure 6.9 – Left anti join

- Right anti join :

A `rightanti` join returns only the rows from the right table that do not have a match in the left table.

Here is an example:

```
Table1
| join kind=rightanti (Table2) on KeyColumn
```

A typical use case would be to find entries in `Table2` that do not have corresponding entries in `Table1`.

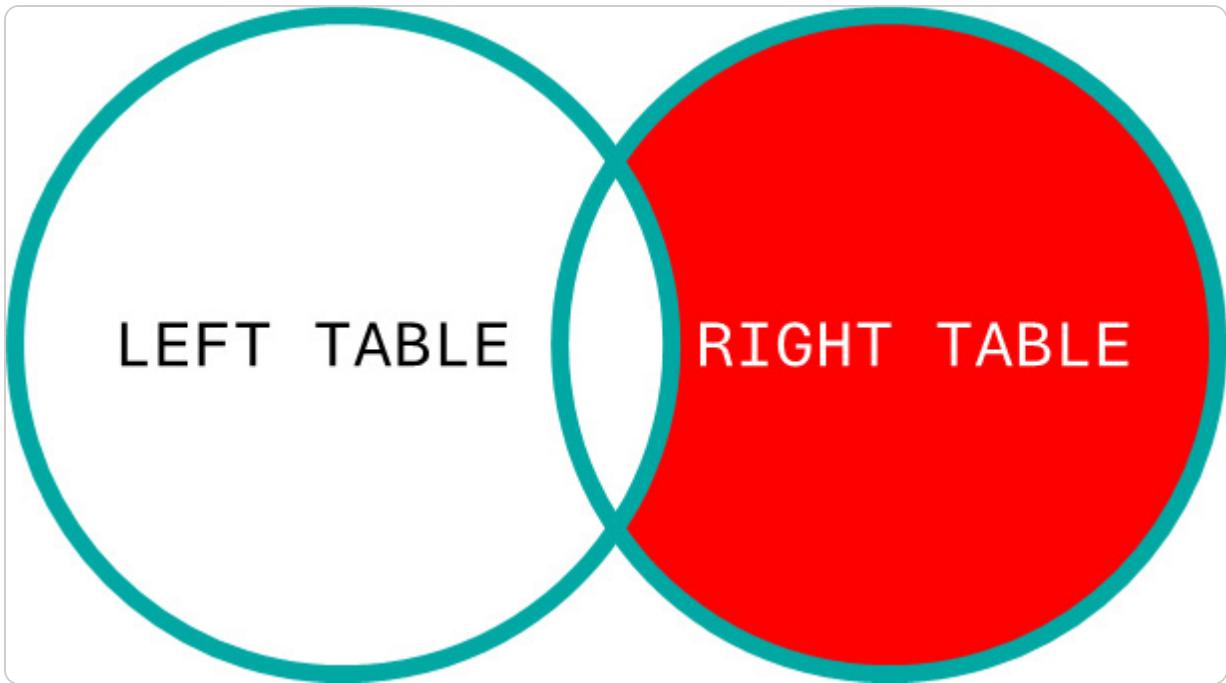


Figure 6.10 – Right anti join

- **Left semi join :**

Like the `leftanti` join, a `leftsemi` join returns all the rows from the left table for which there is at least one match in the right table. Unlike the inner join, it does not return any columns from the right table.

Here is an example:

```
Table1
| join kind=leftsemi (Table2) on KeyColumn
```

These joins are useful for filtering data in the left table based on the existence of matching data in the right table.

- **Right semi join :**

This is the opposite of the `leftsemi` join and returns rows from the right table where corresponding matches exist in the left table, without returning any columns from the left table.

Here is an example:

```
Table1
| join kind=rightsemi (Table2) on KeyColumn
```

These joins are useful when you need to filter rows from the right table based on the presence of matching entries in the left table.

Example scenario

Let's say we want to find users who have logon events but no corresponding security alerts within the last 24 hours. We will use a `leftanti` join to achieve this:

```
IdentityLogonEvents
| where TimeGenerated >= ago(24h)
| join kind=leftanti (
    SecurityAlert
    | where TimeGenerated >= ago(24h)
) on $left.User == $right.User
```

The following list explains the various terms in the preceding code:

- **IdentityLogonEvents** : The table containing logon event data
- **SecurityAlert** : The table containing security alert data
- **leftanti** : This join type returns rows from **IdentityLogonEvents** where there is no matching user in **SecurityAlert**
- **on \$left.User == \$right.User** : The key column to join on, which is **User** in both tables

This query helps identify users who logged on but did not trigger any security alerts, potentially highlighting users who need closer monitoring.

Users who have logged on but have not triggered any security alerts might seem benign at first, but there are several reasons why closer monitoring could be warranted:

- **Potentially suspicious activity :**
 - **New users or devices** : New users or devices logging on without any corresponding alerts could indicate unusual activity that hasn't yet been detected as malicious.
 - **Account compromise** : If an account is compromised, the attacker might try to avoid triggering alerts. The absence of alerts doesn't necessarily mean the absence of malicious activity; it might mean the attacker is being stealthy.
- **Insider threats :**
 - **Malicious insiders** : Users with legitimate access who are involved in malicious activities might try to operate under the radar, avoiding actions that trigger security alerts.
- **Misconfigured alerts :**
 - **Gaps in monitoring** : The absence of alerts could indicate gaps in your alerting configurations. Monitoring these users helps ensure that your security alerting mechanisms are correctly configured and comprehensive.
- **Behavior anomalies :**
 - **Unusual behavior patterns** : Users whose logon activity doesn't align with typical behavior patterns might be worth investigating. For example, logons at odd hours or from unexpected locations.

Imagine you work in an organization where employees typically log in from 8 AM to 6 PM. If you notice a user logging in at 3 AM and no security alerts are triggered, this might warrant further

investigation. It could be legitimate access for maintenance work, or it could be an indicator of compromised credentials.

By understanding these join types, you can effectively combine and analyze data across different tables to gain deeper insights and make informed decisions.

Example scenario 2

We want to find all logon events within the last 24 hours and then join them with high-severity security alerts. This way, we can focus on the most critical incidents that require immediate attention.

We will use the `IdentityLogonEvents` table together with the `SecurityAlert` table to find the most critical logon events. We will use an `inner` join to include only the records that have matching entries in both tables:

```
IdentityLogonEvents
| where TimeGenerated >= ago(24h)
| join kind=inner (
    SecurityAlert
    | where TimeGenerated >= ago(24h)
    | where Severity == "High"
) on $left.User == $right.User
```

Imagine your security team receives numerous alerts daily but you want to prioritize investigating the most severe incidents. By running this query, you can do the following:

- **Identify critical logon events :** Quickly see which user logons are associated with high-severity alerts, indicating potentially serious security threats
- **Focus on severity :** Filter out less critical alerts and focus on those that require immediate action
- **Gain context :** Understand the context around high-severity alerts by viewing related logon events, which can help in assessing the scope and impact of the incident

Let's go through an example scenario in detail:

1. High-severity alerts :

- An attacker might have gained access to a user account and performed suspicious activities
- A critical vulnerability might have been exploited, resulting in high-severity alerts

2. Logon events correlation :

- A high-severity alert is triggered by suspicious login attempts (e.g., multiple failed attempts followed by a successful login from an unusual location)
- The query correlates these alerts with the corresponding logon events, providing a clear picture of what happened

By using the `join` operator with high-severity filters, you can effectively prioritize and investigate critical security incidents. This approach helps ensure that your security team focuses on the most

significant threats, enhancing your organization's ability to respond quickly and effectively to potential breaches.

Grouping and combining data

One of the most essential operators in KQL is the `summarize` operator, which helps us group rows based on the `by` clause into one row for each of the groups.

Let's see how to view logon events by their type and modify your query to group the results. We will also create a new column called `TotalLogins` :

`IdentityLogonEvents`

```
| where Timestamp >= ago(24h)  
| summarize TotalLogins = count() by LogonType
```

In this query, the `summarize` operator is used to count the number of logon sessions based on their type, such as `interactive`, `remote interactive (RDP)`, `network`, `batch`, and `service`. The `by` keyword specifies the field to group by – in this case, `LogonType` – and we display the count of all logins with our new column name, `TotalLogins`.

The output will look like *Figure 6.11*.

Advanced hunting

The screenshot shows the Azure Log Analytics Advanced hunting interface. At the top, there's a navigation bar with 'New query*', a plus sign for creating new queries, and other options like 'Run query', 'Set in query', 'Save', and 'Share link'. Below the navigation is a section titled 'Query' containing the following KQL code:

```
1 IdentityLogonEvents
2 | where Timestamp >= ago(24h)
3 | summarize TotalLogins = count() by LogonType
```

Below the query editor, there are tabs for 'Getting started', 'Results' (which is selected), and 'Query history'. Under 'Results', there are buttons for 'Export' and a search bar with '4 items' results. A 'Filters' button and an 'Add filter' button are also present. The main area displays a table of logon types and their counts:

LogonType	TotalLogins
> Resource access	228
> Interactive	10
> Remote desktop	3
> Failed logon	1

Figure 6.11 – Summarize logon type for the last 24 hours

Creating KQL queries is truly an art form, and as we've explored the essentials – from data management to combining tables – it's crucial to navigate the common pitfalls. Next, we'll dive into practical tips that enhance your ability to craft effective and efficient queries.

Practical tips for effective queries

As you begin crafting your queries, here are some practical tips to keep in mind:

- **Start simple** : Begin with basic queries to get a feel for the data and gradually add complexity.
- **Use intuitive names** : Give meaningful names to your queries and filters to make them easier to understand and maintain.
- **Leverage documentation** : Microsoft provides extensive documentation and examples for KQL, which can be invaluable as you learn.
- **Practice regularly** : The more you use KQL, the more proficient you will become. Regular practice will help you develop a deeper understanding and uncover more advanced techniques.

By mastering these foundational skills, you'll be well-equipped to dive deeper into the world of proactive threat hunting with KQL, enhancing your organization's ability to detect and respond to security threats effectively.

Hunting tables in MDI

To efficiently hunt threats and investigate security incidents, it's crucial to understand the key hunting tables available in MDI. These tables store different types of data collected from your network, which can be queried using KQL to identify potential threats and gain insights into your organization's security posture.

IdentityLogonEvents

The `IdentityLogonEvents` table logs user authentication events, providing detailed information about user logins and logouts in Active Directory, as well as authentication activities in Microsoft Online services monitored by Defender for Cloud Apps. The following are some of the key columns:

- `Timestamp` : The timestamp of the event
- `AccountName` : The user account involved in the event
- `FailureReason` : Information on why the action failed
- `ActionType` : The type of activity that generated the event, such as `LogonFailed` and `LogonSuccess`
- `Protocol` : The protocol used during communication
- `TargetDeviceName` : The **fully qualified domain name (FQDN)** of the device that was impacted by the recorded action
- `AccountUpn` : The user principal name involved in the event
- `Application` : The application that did the action
- `LogonType` : The type of logon (e.g., interactive or network)
- `DeviceName` : The device (FQDN) from which the event originated
- `Location` : The geographic location of the event

NOTE

Only a subset of columns is shown here. Use the schema reference to see all the columns for the table.

This table is essential for tracking user activity, identifying unusual login patterns, and detecting potential unauthorized access.

IdentityDirectoryEvents

The `IdentityDirectoryEvents` table captures directory service-related activities, such as changes to user accounts and group memberships. Here's also where you can find system events on a domain controller such as activities from PowerShell and Task Scheduler. The following are some of the key columns:

- **Timestamp** : The timestamp of the directory event.
- **ActionType** : The type of directory action (for example, Add, Modify, Delete, PowerShell command, WMI execution, etc.) – see here for example:
 - *Account Disabled Changed , Account Password Changed , Device Account Created , Group Membership Changed , Potential lateral movement path identified , Task scheduling , and more*
- **TargetAccountUpn**: The user principal name (UPN) of the account that was affected by the recorded action
- **Protocol**: The protocol used during communication
- **Application** : The application that did the action
- **AccountName** : The user account involved in the event
- **AccountUpn** : The UPN involved in the event
- **DeviceName** : The FQDN of the device

NOTE

Only a subset of columns is shown here. Use the schema reference to see all the columns for the table.

This table is valuable for auditing changes to directory services and identifying potential insider threats or configuration issues.

Start by looking at what `ActionType` you have with the `summarize` operator and the `count()` aggregation function to count the records per summarization group:

```
IdentityDirectoryEvents  
| where Timestamp > ago(7d)  
| summarize count() by ActionType
```

This query gives you a quick snapshot of directory service activities, organized by action type. In the figure below, Advanced Hunting displays these results, highlighting any high-frequency actions that could signal potential risks or unusual configurations.

Advanced hunting

The screenshot shows the Microsoft Sentinel Advanced Hunting interface. At the top, there's a navigation bar with 'New query*' (underlined), a '+' button, a back arrow, 'Run query' (highlighted in blue), a date range 'Last 30 days', 'Save' (with a dropdown arrow), and 'Share link'. Below the navigation is a section titled '^ Query' containing the following PowerShell-like query:

```
1 IdentityDirectoryEvents
2 | summarize count() by ActionType
3 | sort by count_
```

Below the query editor, there are three tabs: 'Getting started', 'Results' (underlined in blue), and 'Query history'. The 'Results' tab displays the output of the query. It includes a 'Export' button, a search bar with a magnifying glass icon, and a note '10 items'. Under 'Filters', there's a 'Add filter' button. The main area shows a table of results:

ActionType	count_
> SMB file copy	251
> Account Password changed	4
> Account Supported Encryption Types changed	2
> Device Account Created	2
> Device dNSHostName changed	2
> SAM Account Name changed	2
> Service creation	2
> Device Operating System changed	1
> Account Name changed	1

Figure 6.12 – Result of the IdentityDirectoryEvents query

The following is a query that will list changes to a specific group (you will now be presented with the `let` statement, which helps us to set a variable name, and, in this scenario, the variable name will be `group`):

```
let group = '<insert your group>';
IdentityDirectoryEvents
| where ActionType == 'Group Membership changed'
| extend AddedToGroup = AdditionalFields['TO.GROUP']
```

```

| extend RemovedFromGroup = AdditionalFields['FROM.GROUP']
| extend TargetAccount = AdditionalFields['TARGET_OBJECT.USER']
| where AddedToGroup == group or RemovedFromGroup == group
| project-reorder Timestamp, ActionType, AddedToGroup, RemovedFromGroup, TargetAccount
| limit 100

```

As you can see, the `AdditionalFields` column has some very informative data for us when we are hunting.

IdentityInfo

This table was renamed from `AccountInfo`. Observe that some of the columns are only visible if the tenant has one of the following licenses: Microsoft Defender for Identity, Microsoft Defender for Cloud Apps, or Microsoft Defender for Endpoint P2 licensing. The following are some of the key columns:

- `Timestamp` : The timestamp of the event
- `AccountObjectId` : Identifier of the account in Entra ID
- `OnpremSid` : SID of the account in Active Directory
- `IsAccountEnabled` : Shows whether the account is enabled or disabled
- `SourceProvider` : Shows the primary identity provider of the account:
 - `AzureActiveDirectory`
 - `ActiveDirectory`
- `RiskLevel` : Shows the risk level (`Low` / `Medium` / `High`) of the user account from Entra ID
- `RiskLevelDetails` : Shows more detail about the risk level from Entra ID

NOTE

Only a subset of columns is shown here. Use the schema reference to see all the columns for the table.

The following is a query that will filter and summarize from the `IdentityInfo` table based on the name of a specific department:

```

let DepartmentName = "<insert your department>";
IdentityInfo
| where Department == DepartmentName
| summarize by AccountObjectId, AccountUpn

```

We start by assigning a new variable called `DepartmentName` with the `let` statement. We then filter the rows with the `where` operator to match the name of the department in the `Department` column, and finally, we summarize the rows with the `summarize` operator by `AccountObjectId` and `AccountUpn`.

In the next query, we will hunt to see whether any user has accessed a specific server and whether that user was not part of the IT department:

```

let LoginEvent = dynamic(["4624", "4672", "4768", "4776"]);
SecurityEvent
| where EventID in (LoginEvent)
| where Computer == "ADFS01.contoso.local"
| join kind=innerunique (
    IdentityInfo
    | summarize arg_max(TimeGenerated, *) by AccountObjectId
    ) on $left.TargetUserSid == $right.AccountSID
| where Department != "IT"

```

At the top of the query, we will yet again start by initiating a new variable with the `let` statement, and now we will have a dynamic list of event IDs that will be used in the `where` operator for the `SecurityEvent` table (which contains security events from Windows machines). We will do another filtering with the `where` operator to target a specific server and, in this case, we are looking at the `ADFS01` server. Next, we will join the `SecurityEvent` table with the `IdentityInfo` table, and the join type is `innerunique`, ensuring that each matching `AccountObjectId` from `IdentityInfo` is linked with the corresponding `TargetUserSid` from the `SecurityEvent` table. The `arg_max` function ensures that only the most recent entry for each user is considered. The final filter removes users belonging to the IT department from the results, focusing the analysis on users from other departments.

IdentityQueryEvents

The `IdentityQueryEvents` table contains data about queries made against the directory service. Here are some of the key columns:

- `Timestamp` : The timestamp of the query event
- `ActionType` : The activity that triggered the event
- `QueryType` : The type of query executed
- `QueryTarget` : The entity (user, group, domain, other) targeted by the query
- `Query` : The string used to run the query

NOTE

Only a subset of columns is shown here. Use the schema reference to see all the columns for the table.

This table is useful for monitoring directory service queries, detecting unusual query patterns, and identifying potential reconnaissance activities.

The following query is designed to identify and analyze potential AS-REP Roasting attacks targeting the `Domain Admins` group. AS-REP Roasting is a technique used by attackers to obtain **ticket granting tickets (TGTs)** for user accounts that have the **Do not require Kerberos preauthentication** attribute set (on the user properties and under the **Account** tab). This query focuses on identifying such attempts within the last 24 hours:

```
IdentityQueryEvents
| where Timestamp > ago(1d)
| where QueryTarget == "Domain Admins"
| where Query contains "attribute"
```

The query filters the `IdentityQueryEvents` table to include events from the past day, specifically those targeting the `Domain Admins` group and containing the word `attribute` in the query. This helps in detecting attempts to enumerate accounts with vulnerable settings.

Practical use of hunting tables

Each of these tables provides a wealth of data that can be queried and analyzed to uncover potential threats. By combining data from multiple tables, security analysts can gain a comprehensive view of user activities, network interactions, and security events. This holistic approach is essential for identifying sophisticated and stealthy threats that may otherwise go unnoticed.

For example, you might use the `SecurityAlert` table to identify high-severity alerts and then correlate this data with logon events from the `IdentityLogonEvents` table to track the involved user's activities. Similarly, network sessions from the `NetworkSession` table can be analyzed alongside device logon events to detect potential lateral movement within your network.

In the following sections, we will explore advanced KQL techniques that leverage these hunting tables to enhance your threat detection and response capabilities, illustrating their practical application through real-world case studies.

Advanced KQL techniques for deep threat detection

In the realm of cybersecurity, **Active Directory (AD)** remains a prime target for attackers seeking to exploit enterprise networks. Understanding known attack paths in Active Directory and leveraging powerful query languages to detect these threats is crucial for defending against sophisticated cyber threats. This section of the chapter delves into advanced KQL techniques for deep threat detection using MDI and Microsoft Defender XDR. We'll start by exploring the basics of common AD attack paths, gradually advancing to complex detection methodologies, and examining how MDI implements detections across various phases of an attacker's kill chain.

Understanding attack paths in AD

AD is a critical component in many organizational IT infrastructures, providing authentication and authorization services. Many IT professionals are saying that AD is a legacy IAM solution, but it is still used at scale and hard to get away from. However, its complexity and centrality also make it a focal

point for attackers. The common attack paths in AD include credential theft, lateral movement, privilege escalation, and persistence mechanisms. Here's a brief overview:

- **Credential theft** : Attackers often use techniques such as phishing, keylogging, or pass-the-hash to steal user credentials. Once acquired, these credentials can be used to move laterally across the network.
- **Lateral movement** : This involves moving from one compromised system to another, often escalating privileges along the way. Tools such as `PsExec` and PowerShell Remoting are frequently used in these attacks.
- **Privilege escalation** : Attackers seek to increase their access within the network. This can involve exploiting vulnerabilities or misconfigurations to gain higher privileges.
- **Persistence** : To maintain access, attackers implant backdoors or manipulate legitimate services to ensure they can return even after detection efforts

MDI and the attacker's kill chain

MDI employs a layered approach to threat detection, addressing each phase of the attacker's kill chain. The kill chain includes reconnaissance, initial access, execution, persistence, privilege escalation, defense evasion, credential access, discovery, lateral movement, collection, command and control, and exfiltration.

MDI detects threats by analyzing signals from various data sources, including security events, network traffic, and behavioral analytics. For each phase of the kill chain, MDI implements specific detections:

- **Reconnaissance** : MDI monitors unusual queries and enumeration activities, indicating an attacker mapping the network
- **Initial access** : Detection focuses on identifying the exploitation of vulnerabilities or the use of stolen credentials
- **Execution** : MDI looks for the execution of malicious scripts or binaries
- **Persistence** : Changes to critical systems or configurations that could allow attackers to maintain access are flagged
- **Privilege escalation** : MDI detects attempts to gain elevated privileges, often through known exploits or credential theft
- **Defense evasion** : Techniques such as disabling security tools or clearing logs are monitored
- **Credential access** : Unusual access patterns to sensitive accounts are analyzed
- **Discovery and lateral movement** : MDI analyzes lateral movement techniques and suspicious network traffic patterns
- **Collection and exfiltration** : Anomalous data access and transfer activities are detected to prevent data theft

As of the time of writing this book, there were over 70 documented detections available on Microsoft Learn. These detections cover a wide range of attack techniques and scenarios, providing valuable guidance on what types of activities should be monitored to protect your environment. However, you will not find the exact KQL queries for these detections directly documented. This is primarily due to security reasons and the proprietary nature of Microsoft's detection methodologies. By not disclosing the specific queries, Microsoft aims to prevent attackers from reverse-engineering the detection logic to evade detection. Instead, Microsoft provides high-level descriptions and guidelines, allowing security professionals to understand the nature of the threats and the underlying principles of the

detections. This approach ensures that while defenders are equipped with the knowledge to protect their environments, attackers are not given a roadmap to bypass these defenses. For further reference, Microsoft provides categorized documentation on various types of security alerts that MDI can generate. These resources offer insight into the specific activities and techniques that should be monitored within each category:

- *Reconnaissance and discovery alerts* : <https://learn.microsoft.com/en-us/defender-for-identity/reconnaissance-discovery-alerts>
- *Persistence and privilege escalation alerts* : <https://learn.microsoft.com/en-us/defender-for-identity/persistence-privilege-escalation-alerts>
- *Credential access alerts* : <https://learn.microsoft.com/en-us/defender-for-identity/credential-access-alerts>
- *Lateral movement alerts* : <https://learn.microsoft.com/en-us/defender-for-identity/lateral-movement-alerts>
- *Other security alerts* : <https://learn.microsoft.com/en-us/defender-for-identity/other-alerts>

Crafting KQL queries for threat detection

As you may have understood at this point, KQL is a robust tool in your arsenal for efficiently detecting and analyzing threats within your on-premises and cloud environment. In this section, we'll delve into several practical examples to help you leverage KQL to enhance your security operations.

Example 1 – Identifying NTLM and Kerberos sign-ins

Monitoring the authentication protocols used in your environment is crucial for identifying potential security issues. **New Technology LAN Manager (NTLM)** is an older and less secure protocol compared to Kerberos. Therefore, understanding the ratio of NTLM to Kerberos sign-ins can highlight areas where security improvements are needed. Looking at `IdentityLogonEvents` and summarizing the rows by `Protocol` gives us a hint of whether we need to change the authentication protocols in our environment:

```
IdentityLogonEvents
| where Application == "Active Directory"
| where Protocol in ("Ntlm", "Kerberos")
| where ActionType == "LogonSuccess"
| summarize count() by Protocol
| render piechart
```

Let's look at this in more detail:

- `IdentityLogonEvents` : This table logs all identity-related events
- `Application == "Active Directory"` : Filters events to only those related to Active Directory
- `Protocol in ("Ntlm", "Kerberos")` : Selects logon events that used either NTLM or Kerberos
- `ActionType == "LogonSuccess"` : Focuses on successful logon attempts
- `summarize count() by Protocol` : Aggregates the count of logons by protocol
- `render piechart` : Visualizes the data as a pie chart to easily compare the usage of NTLM versus Kerberos

Example 2 – Monitoring legacy service accounts

While **Group Managed Service Accounts (gMSAs)** are recommended, legacy service accounts still exist and can pose a security risk if not properly monitored. It's crucial to track their activity to detect any unusual behavior:

```
IdentityLogonEvents
| where Application == "Active Directory"
| where AccountUpn == "your_svcaccount@domain.local" // Edit to your SVC account
| where ActionType == "LogonSuccess"
| summarize count() by DeviceName
```

Let's look at this in more detail:

- `AccountUpn == "your_svcaccount@domain.local"` : Filters events for a specific service account. Replace `your_svcaccount@domain.local` with the actual UPN of your service account.
- `summarize count() by DeviceName` : Counts the number of successful logons by device, helping you understand where this service account is being used.

Example 3 – Identifying all service accounts

If you're unsure about all the service accounts in your environment, you can query the `IdentityInfo` table to get a comprehensive list:

```
IdentityInfo
| where SourceProvider == "ActiveDirectory"
| where Type == "ServiceAccount"
| summarize arg_max(AccountName, *) by Timestamp
| sort by Timestamp desc
```

Let's look at this in more detail:

- `SourceProvider == "ActiveDirectory"` : Filters for entries sourced from AD
- `Type == "ServiceAccount"` : Ensures only service accounts are included
- `summarize arg_max (Accpint Name, *) by Timestamp` : Aggregates the latest information about each service account based on the timestamp
- `sort by Timestamp desc` : Sorts the results by the most recent entries, providing an up-to-date view of service accounts

Example 4 – Monitoring multiple service accounts

To monitor multiple service accounts simultaneously, you can create a dynamic list of service accounts and use it in your query:

```
let srvclist =
dynamic(["svc1@contoso.local", "svc2@contoso.local", "svc3@contoso.local"]);
IdentityLogonEvents
| where AccountUpn in~ (srvclist)
| summarize count() by AccountName, DeviceName, Protocol
```

Let's look at this in more detail:

- `let srvcList = dynamic([...])` : Creates a dynamic list of service account UPNs
- `AccountUpn in~ (srvcList)` : Filters logon events to include only those from the specified service accounts
- `summarize count() by AccountName, DeviceName, Protocol` : Aggregates the logon events by account name, device, and protocol to provide insights into where and how these service accounts are being used

Example 5 – Ensuring domain controllers are not used as file servers

Using domain controllers as file servers can pose significant security risks. It's essential to ensure that this practice is not happening in your environment:

```
IdentityDirectoryEvents
| where ActionType == "SMB file copy"
| extend ParsedFields=parse_json(AdditionalFields)
| extend FileName=tostring(ParsedFields.FileName),
FilePath=tostring(ParsedFields.FilePath), Method=tostring(ParsedFields.Method)
| where Method == "Write"
| project Timestamp, ActionType, DeviceName, IPAddress, AccountDisplayName,
DestinationDeviceName, DestinationPort, FileName, FilePath, Method
```

Let's look at this in more detail:

- `ActionType == "SMB file copy"` : Filters for events related to SMB file copy actions
- `extend ParsedFields = parse_json(AdditionalFields)` : Parses additional fields stored as JSON and saves it to a new column named `ParsedFields`
- `extend FileName = ..., FilePath = ..., Method = ...` : Extracts specific fields from the parsed JSON
- `where Method == "Write"` : Focuses on write operations, indicating files being copied to the server
- `project ...` : Selects and displays relevant fields for further analysis

Example 6 – Detecting enumeration attacks

Enumeration attacks can be detected by monitoring query events against your Active Directory. These attacks often involve using Security Account Manager (SAM) Remote Protocol or Lightweight Directory Access Protocol queries to gather information about users and groups:

```
IdentityQueryEvents
| where Application == "Active Directory"
| where ActionType in ("SAMR", "LDAP")
| project Timestamp, ActionType, DeviceName, DestinationDeviceName,
AccountDisplayName, QueryType, QueryTarget
```

Let's look at this in more detail:

- `IdentityQueryEvents` : This table logs queries made against the directory
- `Application == "Active Directory"` : Filters for events specific to Active Directory
- `ActionType in ("SAMR", "LDAP")` : Selects SAMR and LDAP query types, which are commonly used in enumeration attacks
- `project ...` : Projects specific fields to help identify which devices and accounts are making these queries and what they are querying for

By leveraging these KQL queries, you can enhance your threat detection capabilities and maintain a secure AD environment. Whether you are new to protecting AD or an experienced SecOps professional, these examples provide valuable insights and practical techniques for effective security monitoring.

In the next section, we will explore three common and impactful AD attacks: **pass-the-hash** (PtH), Kerberoasting, and DCShadow.

The **PtH attack** occurs during the lateral movement phase of the kill chain, where attackers use hashed credentials to move within a network without needing plaintext passwords. **Kerberoasting** targets the credential access phase by requesting service tickets for service accounts and cracking them offline to retrieve plaintext passwords. Lastly, the **DCShadow attack** impacts the persistence and privilege escalation phases, as attackers register rogue domain controllers to inject malicious changes into Active Directory.

Real-world case studies – detecting advanced attacks with KQL

In this section, we will delve into the fascinating world of cyber defense by exploring three sophisticated AD attacks. Using KQL with MDI and MDE we will uncover the techniques used to detect these stealthy threats. Through these real-world case studies, you'll gain a deep understanding of how to identify and mitigate these advanced attacks, enhancing your organization's security posture.

The attacks we'll cover include the following:

- **PtH attack** : Exploiting hashed credentials to gain unauthorized access
- **Kerberoasting** : Targeting service accounts to crack encrypted tickets and retrieve plaintext passwords
- **DCShadow attack** : Registering rogue domain controllers to push malicious changes into the Active Directory

By the end of this section, you'll not only learn how these attacks work but also how to craft and apply advanced KQL queries to detect and respond to these threats effectively. Let's dive into the practical application of these detection techniques and see how they play out in real-world scenarios.

Prerequisites

If you want to follow along in your own environment, you can use the Bicep template provided on GitHub to deploy an Active Directory environment in your Azure subscription. The code is located at <https://github.com/PacktPublishing/Microsoft-Defender-for-Identity-in-Depth/tree/main/Chapter06>.

Verify that you have the following resources:

- Windows Server with Active Directory installed

- Windows workstation and/or server: Joined to the domain
- User accounts: Create several user accounts and service accounts with Service Principal Names (SPNs)
- Attack tools: Download and install the necessary tools such as Mimikatz

Installing Active Directory Domain Services

If Active Directory Domain Services is not already installed on your Windows Server, proceed with the installation and promote the server to a domain controller. Create some user accounts, including a backup Domain Admin account, to ensure you can access your domain controller after the upcoming lab.

Creating standard user accounts and service accounts

We'll start by creating the necessary standard user accounts and service accounts using the following steps:

1. Define the variables:

```
$serviceAccountName = "svc_kerberoast"
$domain = "contoso.local"
$serviceAccountPassword = "P@ssw0rd!" # Only used for lab purposes
$spn = "HTTP/webserver.contoso.local"
$ou = "OU=Service Accounts,DC=contoso,DC=local" # Change this to the desired OU
```

2. Import the Active Directory module:

```
Import-Module ActiveDirectory
```

3. Create the service account:

```
New-ADUser -Name $serviceAccountName -SamAccountName $serviceAccountName -
UserPrincipalName "$serviceAccountName@$domain" -Path $ou -AccountPassword
(ConvertTo-SecureString $serviceAccountPassword -AsPlainText -Force) -Enabled
$true
```

4. Set the SPN for the service account:

```
Set-ADUser -Identity $serviceAccountName -ServicePrincipalNames @{Add=$spn}
```

5. Verify that the SPN has been set:

```
Get-ADUser -Identity $serviceAccountName -Property ServicePrincipalName | Select-
Object -ExpandProperty ServicePrincipalName
```

Joining workstations to the domain

Make sure that you have the correct IP address, subnet mask, default gateway, and DNS settings. Prerequisites are workstations and joining to the domain on your network adapter. Also, make sure that you have your Domain Admin credentials for the domain join process.

On your Windows workstation/server, join the domain through **System Properties > Change settings > Domain**.

Or in an elevated PowerShell window, type `sconfig` and then follow this guide to start an elevated PowerShell window.

Click on the **Start** menu, type `PowerShell` in the search bar, right-click on **Windows PowerShell**, and select **Run as administrator**.

Installing Mimikatz

Please note that installing Mimikatz is a prerequisite. Mimikatz is a powerful post-exploitation tool often used for legitimate security testing and research. It should be used responsibly and only in environments where you have explicit permission to do so.

Here are some important considerations:

- **Ethical use** : Ensure you have explicit permission to use Mimikatz on any network or system. Unauthorized use is illegal and unethical.
- **Security** : Use Mimikatz in a controlled environment to avoid accidental data leakage or exposure.
- **Documentation** : Keep detailed logs of your activities when using Mimikatz for security testing or research purposes.

Step 1 – Disable MDE antivirus temporarily

Many antivirus programs detect Mimikatz as a malicious tool due to its capabilities. If you are using this for legitimate purposes, you may need to temporarily disable your antivirus software. Follow these steps to disable MDE antivirus:

1. Log in to the Defender XDR portal (<https://security.microsoft.com>) with an account that has **Manage Security** settings such as the *Security Administrator* role.
2. In the Defender XDR menu, click on **Assets > Devices**.
3. Search for and find your machine in **Device Inventory** that we will be doing the tests against.
4. Click on the ellipsis (more actions) at the top right of the page.
5. Click on **Turn on troubleshooting mode**.
6. Click on **Submit** to start the troubleshooting mode.
7. Log in to your machine and follow the rest of the guide.
8. Click on the **Start** menu, type `PowerShell` in the search bar, right-click on **Windows PowerShell**, and select **Run as administrator**.
9. Type the following commands to disable MDE:

```
Set-MpPreference -DisableIntrusionPreventionSystem $true -DisableIOAVProtection  
$true -DisableRealtimeMonitoring $true -DisableScriptScanning $true -  
EnableControlledFolderAccess Disabled -EnableNetworkProtection AuditMode -Force -  
MAPSReporting Disabled -SubmitSamplesConsent NeverSend
```

10. Verify that the preceding settings are disabled with the following command:

```
Get-MpPreference
```

11. Confirm the action and remember to re-enable your antivirus once you are done using Mimikatz. Remember that MDI and some MDE functionality are expected in this lab.

Step 2 – Download Mimikatz

1. Open a web browser and navigate to the official Mimikatz GitHub repository: <https://github.com/gentilkiwi/mimikatz/releases>.
2. Download the latest compiled version of Mimikatz. Look for a file named something like **mimikatz_trunk.zip**. You might see warnings from Defender SmartScreen. You can disable SmartScreen if you wish; otherwise, click on **Keep file** a few times.

Step 3 – Extract the files

1. Once downloaded, navigate to your **Downloads** folder and locate the **mimikatz_trunk.zip** file.
2. Right-click on the file and select **Extract All...**.
3. Choose a destination folder and click **Extract**. This will create a folder with the Mimikatz executable and other files.

Step 4 – Run Mimikatz

1. Open the folder where you extracted Mimikatz.
2. Right-click on **mimikatz.exe** and select **Run as administrator**. This is important because Mimikatz requires administrative privileges to perform its tasks.
3. A Command Prompt window will open with the Mimikatz interface. We will use this Command Prompt later.

Deploy Microsoft Defender

To enhance your network's security posture, it's critical to properly deploy Microsoft Defender components. The following are detailed steps for setting up Defender for Identity and Defender for Endpoint/Servers, ensuring comprehensive protection across different environments:

- **Defender for Identity** : Install sensors on DCs, federation servers (AD FS), certificate servers (AD CS), and Entra Connect servers, following the second chapter of this book. Make sure that you don't have any health issues for the sensor.
- **Defender for Endpoint/Defender for Servers** :
 - For workstations, deploy **Defender for Endpoint**, following Microsoft's documentation:
<https://docs.microsoft.com/en-us/microsoft-365/security/defender-endpoint/>
 - For servers in Azure, deploy **Defender for Servers** through Defender for Cloud:
 - i. Log in to <https://portal.azure.com> with an account that has either a Security Administrator role (from Entra ID), Contributor, or Owner (via Azure RBAC on subscription level).
 - ii. Search for and select **Microsoft Defender for Cloud**.
 - iii. In the **Defender for Cloud** menu, select **Environment settings**.
 - iv. Select the subscription or workspace that you want to protect.
 - v. Under **Cloud Workload Protection (CWP)**, select the **On** setting for the **Servers** plan.
 - vi. Under the **Monitoring Coverage** column, click on **Settings**.
 - vii. Make sure that **Endpoint protection** has the status of **On**.

viii. Click on **Continue** and then **Save** at the top of the page.

Defender XDR advanced hunting

With your favorite web browser, access the Microsoft Defender XDR portal at <https://security.microsoft.com>, and in the menu, navigate to **Investigation & response > Hunting > Advanced Hunting**.

Use the provided KQL queries in the following sections to monitor and detect suspicious activities.

PtH attack

A PtH attack allows an attacker to authenticate as a user by using the hashed value of the user's password, instead of the plaintext password. This type of attack exploits the way Windows systems handle authentication, specifically the NTLM authentication protocol.

NTLM DEPRECATED

In June 2024, Microsoft announced that NTLM (LANMAN, NTLMv1, and NTLMv2) are no longer under active development and are deprecated. It's highly recommended to transition from NTLM to more secure authentication methods. NTLM will continue to function in the upcoming releases of Windows Server and the annual releases of Windows. However, calls to NTLM should be replaced with calls to Negotiate, which will attempt to authenticate using Kerberos and only revert to NTLM when necessary.

How it works

Now that we have a clear understanding of what a PtH attack involves and the implications of NTLM deprecation, let's delve into the specifics of how these attacks are carried out. The process typically starts with obtaining the necessary credentials, followed by unauthorized access using those credentials:

1. **Obtaining the hash** : The attacker first needs to obtain the hashed password (NTLM hash) of a target user. This can be achieved through various methods, such as extracting hashes from a compromised machine's memory, **Security Account Manager (SAM)** database, or an **NTDS.dit** file (Active Directory database) using tools such as Mimikatz.
2. **Using the hash for authentication** : Once the attacker has the hash, they can use it to authenticate to other machines on the network without needing the user's plaintext password. The hash acts as a substitute for the password, enabling the attacker to move laterally across the network.

Attack vector

The PtH attack is particularly potent because it allows attackers to move laterally within a network and escalate their privileges. By exploiting the NTLM authentication mechanism, attackers can gain unauthorized access to other systems and sensitive resources, making it a significant threat:

- **Lateral movement** : PtH is often used for lateral movement within a network. After compromising one machine and extracting password hashes, attackers can use these hashes to access other machines, escalating their privileges and gaining control over

more critical systems

- **Privilege escalation :** Attackers can leverage PtH attacks to gain administrative privileges on other machines by using hashes obtained from compromised admin accounts

Mitigation and detection strategies

To mitigate the risks associated with PtH attacks, it's crucial to adopt more secure practices and technologies. Encouraging the use of Kerberos authentication over NTLM is a vital step, as Kerberos is more secure and less vulnerable to such attacks. Additionally, enabling Windows Defender Credential Guard can provide enhanced protection by isolating credentials so that only privileged system software can access them.

Implementing **multi-factor authentication (MFA)** adds an extra layer of security, making it significantly more challenging for attackers to exploit stolen hashes. Regularly updating and patching systems is essential to protect against known vulnerabilities that could be used to obtain hashes.

Network segmentation plays a key role in limiting lateral movement within the network, thereby restricting the potential damage an attacker can cause. Finally, regular monitoring and auditing of authentication logs are critical to detect unusual login activities that might indicate PtH attacks.

Detecting PtH attacks involves being vigilant for unusual authentication patterns and behaviors. This includes multiple logon attempts using NTLM hashes, authentication from unexpected sources or locations, and the use of administrative accounts in atypical ways.

Steps to perform a PtH attack

The following steps will guide you through executing a PtH attack, which involves obtaining and using hashed credentials to authenticate within a network without needing plaintext passwords:

1. Obtain NTLM hashes :

- On a compromised machine, run Mimikatz to extract NTLM hashes.
- Execute the following commands in Mimikatz:

```
privilege::debug  
sekurlsa::logonpasswords
```

- Note the NTLM hash of a user account.

2. Use NTLM hash to authenticate :

- Use the NTLM hash to authenticate to another machine in the network using Mimikatz:

```
sekurlsa::pth /user:<username> /domain:<domain> /ntlm:<ntlm_hash>
```

3. Verify access:

- A new Command Prompt opens with the privileges of the target user. Try accessing network resources to verify the attack.

Detection

We can hunt for PtH and Mimikatz commands in the `DeviceEvents` table:

```
DeviceEvents
| extend AdditionalFieldsParsed = parse_json(AdditionalFields)
| extend Description = tostring(AdditionalFieldsParsed.Description)
| where Description has "sekurlsa::pth"
| project-reorder Timestamp, Description, InitiatingProcessFolderPath,
InitiatingProcessAccountName
```

Kerberoasting

Kerberoasting is an attack against the Kerberos authentication protocol where attackers exploit the way service accounts handle service tickets. By requesting service tickets for service accounts, attackers can extract these tickets and attempt to crack them offline to retrieve the plaintext passwords. To understand Kerberoasting in action, here's a breakdown of the typical steps attackers use to exploit service accounts:

How it works

1. **Enumerating service accounts** : Attackers start by identifying service accounts in the domain that have **service principal names** (SPNs). SPNs are unique identifiers associated with service instances, and they are necessary for Kerberos authentication.
2. **Requesting service tickets** : Once the service accounts with SPNs are identified, the attacker requests a **ticket-granting service** (TGS) ticket for these accounts. The Domain Controller provides the requested service tickets, encrypted with the service account's password hash.
3. **Extracting and cracking tickets** : The attacker then extracts these service tickets from memory and saves them for offline cracking. Tools such as Rubeus, Invoke-Kerberoast, and Mimikatz can be used for this purpose. The extracted tickets are then subjected to brute-force or dictionary attacks using tools such as Hashcat or John the Ripper to reveal the plaintext passwords.

Attack vector

Kerberoasting is a powerful attack vector because it targets service accounts, which often have elevated privileges and long-lived passwords. By exploiting the Kerberos authentication protocol, attackers can extract and crack service tickets to gain unauthorized access and potentially escalate their privileges within the network:

- **Targeting service accounts** : Service accounts often have higher privileges and long-lived passwords, making them attractive targets for attackers
- **Privilege escalation** : Gaining access to service accounts can allow attackers to escalate their privileges and move laterally within the network

Mitigation and detection strategies

To mitigate the risks associated with Kerberoasting attacks, it's crucial to adopt a combination of preventive measures and vigilant monitoring. First, ensure that service account passwords are strong and complex, making brute-force attacks significantly more difficult. Regularly rotating these passwords can further reduce the risk of password compromise.

Implementing **Managed Service Accounts (MSAs)** or gMSAs can help, as they automatically manage and rotate passwords, enhancing security without manual intervention. Adhering to the principle of least privilege is also important; service accounts should only have the permissions necessary to perform their tasks, limiting potential damage if compromised.

Regular monitoring and auditing of Kerberos ticket requests and authentication logs are essential. This can help detect unusual activities that might indicate a Kerberoasting attempt.

Detecting Kerberoasting involves keeping an eye on abnormal Kerberos ticket requests and activities. Be on the lookout for a high volume of TGS requests, especially for service accounts. TGS requests from non-privileged users and unusual patterns or times of ticket requests are also red flags.

Regularly reviewing logs and setting up alerts for such activities can help in early detection and mitigation of Kerberoasting attacks.

Steps to perform Kerberoasting

The following steps will demonstrate how to carry out a Kerberoasting attack, which involves requesting service tickets for service accounts and attempting to crack them offline to reveal plaintext passwords:

1. **Enumerate service accounts with SPNs :**

- Use PowerShell to list service accounts with SPNs:

```
Get-ADUser -Filter {ServicePrincipalName -ne "$null"} -Properties  
ServicePrincipalName | Select-Object SamAccountName, ServicePrincipalName
```

2. **Request service tickets (TGS) for these SPNs :**

- Use **Rubeus** or **Invoke-Kerberoast** to request service tickets:

```
Invoke-Kerberoast -OutputFormat Hashcat  
Rubeus.exe kerberoast
```

3. **Extract and crack tickets :**

- Extract the tickets and save them to a file.
- Use a cracking tool such as Hashcat to crack the hashes:

```
hashcat -m 13100 -a 0 <hash_file> <wordlist>
```

Detection

As you can imagine, we do need many types of logs to be able to detect malicious events in our environment. Sending events from the security event log is crucial. As you may see in your production environment, there will be a lot of requests for service tickets, but to give you a start, here's a detection you can start with:

```
SecurityEvent
| where EventID == 4769
| where AccountName !endswith "$" // Filter out service or machine accounts
| where ServiceName !endswith "$" // Filter out service accounts
| where TicketEncryptionType == "0x17" // Focus on tickets using RC4 encryption
| project TimeGenerated, AccountName, ServiceName, ClientAddress, TicketEncryptionType
```

DCShadow attack

DCShadow is an attack where an attacker registers a rogue **domain controller (DC)** and uses it to push malicious changes into AD. This attack exploits the replication process of AD, allowing attackers to introduce unauthorized changes that can be difficult to detect.

How it works

The DCShadow attack follows a series of steps that allow attackers to exploit AD replication and make stealthy changes. Here's an outline of how the attack typically unfolds:

1. **Compromise a domain admin account** : The attacker first needs to obtain administrative privileges on the domain, typically by compromising a Domain Admin account.
2. **Register a rogue DC** : Using tools such as Mimikatz, the attacker registers a rogue DC within the AD environment. This rogue DC can then participate in the AD replication process.
3. **Push malicious changes** : The attacker uses the rogue DC to replicate malicious changes into the AD schema or configuration. These changes can include adding new user accounts with elevated privileges, modifying group memberships, or altering security policies.

Attack vector

The DCShadow attack is particularly dangerous because it leverages the inherent trust within the AD replication process. By registering a rogue DC, attackers can introduce malicious changes that propagate throughout the AD infrastructure. This attack vector allows for persistent and stealthy modifications that can go undetected for extended periods. The ability to alter AD schema and security configurations provides attackers with significant control over the network environment, enabling them to escalate privileges, maintain long-term access, and disrupt security policies. This makes DCShadow a powerful tool in an attacker's arsenal for compromising enterprise networks:

- **Persistence and stealth** : By leveraging the replication process, attackers can introduce persistent changes that are difficult to detect and can survive system reboots
- **Privilege escalation** : Attackers can use DCShadow to escalate privileges by modifying user accounts and group memberships

- **Impact on security policies** : Malicious changes to security policies can weaken the overall security posture of the organization, making it easier for attackers to maintain access and move laterally within the network

Mitigation and detection strategies

Mitigating the risks associated with DCShadow attacks requires a combination of robust monitoring, secure administrative practices, and proactive defense measures. Monitoring for unusual replication traffic is crucial, as it can help identify rogue DCs attempting to register and propagate malicious changes. Implementing an Active Directory tiering model helps limit the exposure of high-privilege accounts, making it more difficult for attackers to compromise critical administrative credentials.

Enabling and regularly reviewing Active Directory auditing logs allows for tracking changes in domain controller registrations and modifications to AD objects. Utilizing **secure admin workstations (SAWs)** or **privileged access workstations (PAWs)** for administrative tasks can further reduce the risk of credential compromise by isolating administrative activities on hardened devices. Regularly rotating administrative credentials ensures that even if an account is compromised, its utility for attackers is limited over time.

Detecting DCShadow attacks involves vigilant monitoring for abnormal AD replication activities and changes to domain controller configurations. Key indicators include unexpected domain controller registrations, unusual replication traffic patterns, and modifications to critical AD objects or security policies. Setting up alerts for these activities and regularly reviewing logs can aid in the early detection and mitigation of DCShadow attacks.

By combining these mitigation and detection strategies, organizations can enhance their defense against DCShadow attacks and protect the integrity of their Active Directory environment.

Steps to perform a DCShadow attack

The following steps outline the process of performing a DCShadow attack, where an attacker registers a rogue domain controller and pushes malicious changes into the Active Directory through the replication process:

1. Switch to SYSTEM :

- Run the following commands in Mimikatz. You will see in the output that you are running in **SYSTEM** context:

```
privilege::debug  
token::elevate
```

2. List all tokens and find the Domain Admin account's token ID :

- Start listing all tokens and remember the ID of a Domain Admin account; we will use that ID later:

```
token::list
```

3. Register a rogue DC using Mimikatz :

- Use Mimikatz to register a rogue change in AD. In the following example, we are changing the description of the object in AD:

```
lsadump::dcshadow /object:CN=Administrator,CN=Users,DC=domain,DC=com  
/attribute:description /value:"DCShadow Test"
```

4. Push malicious changes to AD :

- Pushing the changes to AD requires a replication; therefore, we need to open a new Mimikatz Command Prompt as our compromised Domain Admin account. For example, push the description changes that we did earlier:

```
lsadump::dcshadow /push
```

Detection

Due to the sophisticated nature of DCShadow, we may need to employ a variety of KQL queries to effectively monitor for signs of this attack. The following is an example that concentrates on tracking event logs for any unusual domain controller activities:

```
let startTime = ago(1d);  
let endTime = now();  
SecurityEvent  
| where TimeGenerated between (startTime .. endTime)  
| where EventID in (4742, 4662, 4929, 4931)  
| where AccountName !contains "$" // Filtering out service accounts generally  
| extend ObjectModified = tostring(TargetObject)  
| where ObjectModified contains "CN=Domain Controllers"  
| project TimeGenerated, EventID, AccountName, ComputerName, ObjectModified, Activity  
| distinct TimeGenerated, AccountName, ComputerName, ObjectModified, EventID, Activity
```

Good job! Keep on learning and happy hunting!

Summary

In this chapter, we explored the fundamentals of KQL syntax and its bids in threat hunting. We delved into the various MDI tables and highlighted the importance of integrating additional tables to effectively correlate and identify suspicious events. Through hands-on examples, we conducted basic AD attacks to observe how MDI, in conjunction with other Defender products, responds to these threats.

Now, we will focus on taking decisive actions on the incidents and alerts we have identified. The upcoming chapter will guide you through the processes of investigation and response, equipping you with the tools and knowledge to manage and mitigate security threats proactively and efficiently.

Further reading

- *The Definitive Guide to KQL: Using Kusto Query Language for operations, defending, and threat hunting –*
<https://aka.ms/KQLMSPress/Store>

Part 3: Operational Excellence with Microsoft Defender for Identity

This part focuses on achieving operational excellence with **Microsoft Defender for Identity (MDI)**. You'll learn how to investigate and respond to security alerts efficiently, manage MDI action accounts, and build a robust framework for identity threat detection and response. The chapters also address troubleshooting and optimization strategies, ensuring that your MDI deployment remains effective and resilient in the face of evolving threats.

This part includes the following chapters:

- [Chapter 7](#), *Investigating and Responding to Security Alerts*
- [Chapter 8](#), *Utilizing MDI Action Accounts Effectively*
- [Chapter 9](#), *Building a Resilient Identity Threat Detection and Response Framework*
- [Chapter 10](#), *Navigating Challenges: MDI Troubleshooting and Optimization*

7

Investigating and Responding to Security Alerts

In this chapter, we focus on the crucial processes of investigating and responding to security alerts within **Microsoft Defender for Identity (MDI)**. Our journey begins with establishing a methodical approach for effective alert investigation, ensuring that threats are identified and assessed accurately. This foundation is essential for leveraging MDI's capabilities in your security operations.

As we advance, we present a real-world playbook for responding to advanced threats. This section will detail strategies and steps for swift and decisive action, equipping you with the skills to handle complex security challenges. You will learn how to implement response strategies that contain and eradicate threats efficiently.

Finally, the chapter outlines a comprehensive incident response plan for high-stakes situations.

Preparing organizations to manage and mitigate potential security incidents effectively, this section covers the creation of a structured response plan, coordination with stakeholders, and post-incident activities.

By the end of this chapter, you will have a comprehensive understanding of how to investigate and respond to security alerts using MDI. You will be equipped to handle advanced threats and high-stakes incidents, improving your organization's security posture and resilience.

In this chapter, we're going to cover the following main topics:

- Developing a methodical approach to alert investigation
- Real-world playbook: responding to advanced threats
- Incident response- an action plan for high-stakes situations

Developing a methodical approach to alert investigation

In today's cybersecurity landscape, having a structured and systematic approach to investigating security alerts is critical. Developing a methodical approach to alert investigation ensures that security teams can accurately identify, assess, and respond to potential threats, minimizing the risk of breaches and reducing the time to remediation. This section explores the essential steps and best practices for creating a robust investigation process within MDI and **Defender XDR**. By leveraging MDI's powerful features and integrating them into a cohesive strategy, organizations can enhance their threat detection capabilities and ensure a proactive defense posture.

Understanding the MDI alert system

The MDI alert system is designed to detect and notify security teams of potential threats within an organization's network. Understanding how this system works is crucial for effectively managing and responding to MDI-based alerts.

In the **Microsoft Defender** product line, alerts are generated whenever a threat is detected by the respective Defender product. These alerts and the associated Defender product are then indicated in the **Detection source** column within the Defender XDR portal for the alert or incident. When multiple alerts share common characteristics, such as similar attributes or attack techniques, they are aggregated into a single incident to streamline investigation and response efforts.

Each alert and incident is assigned a **severity level**, accompanied by a corresponding color to help visualize the threat's urgency. The severity levels range from the lowest, **Informational**, to **Low**, **Medium**, and the highest, **High**. This grading system aids security teams in prioritizing their responses, ensuring that the most critical threats are addressed promptly. And as you see in *Figure 7.1*, we are presented with the colors, from gray to dark red.

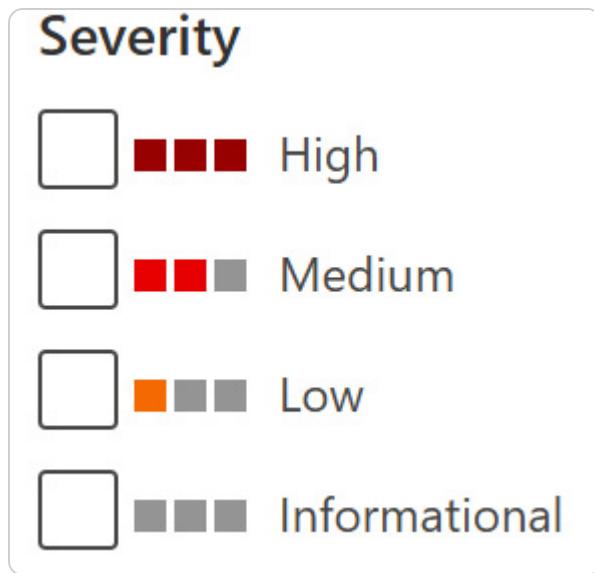


Figure 7.1 – Severity levels of alerts and incidents

As we learned in the previous chapter, MDI has at the time of writing this book, a robust catalog of over 70 distinct alerts. These alerts are designed to detect a wide range of suspicious activities and potential security threats, providing wide-ranging coverage to protect your organization's infrastructure. Each alert is meticulously crafted to address specific scenarios and threat vectors, ensuring that SecOps teams are quickly notified of any unusual or malicious behavior.

Read more about each MDI alert at <https://learn.microsoft.com/en-us/defender-for-identity/alerts-overview>.

IMPORTANT

Some of the built-in alerts in MDI are only supported by using the full installation of the MDI sensor on domain controllers, AD CS, AD FS, and Entra Connect servers. To ensure full coverage and optimal detection capabilities, make sure to deploy the MDI sensor comprehensively across all relevant servers.

When you access the alert or incident within the Defender XDR portal, you will see a great visualization of what occurred as a story, which you will see under the **Alert story** section, see *Figure 7.2*. We will also be presented with the entities involved and details about the activities that occurred. For an analyst to gather more intel about the related entities, it is crucial to make appropriate decisions about whether a device needs to be isolated, or if a user needs to change the password and get the sessions invoked, or whether to gather more information about the IP address.

The screenshot shows the Microsoft Defender XDR portal interface. At the top, there's a navigation bar with icons for Home, Threats, and Alerts, followed by a search bar and a PT (Pilot Test) button. The main area is titled "Alerts > Suspected DCShadow attack (domain controller replication request)". A modal window is open, showing two hosts: "DC01" (Destination Host) and "WIN-GE3SQT7IIQE" (Source Host). Below this, there's an "Alert story" section with a timeline diagram. The timeline shows "WIN-GE3SQT7IIQE" sending "changes to directory objects" to "DC01". The "What happened" section notes that "WIN-GE3SQT7IIQE, which is not a valid domain controller in , sent changes to directory objects on DC01.". The "Alert graph" section provides a visual representation of the interaction. On the right side of the modal, there's an "IN INSIGHT" section with a "Classify alert" button, and an "Alert state" section where "Classification" is set to "Not Set" and "Assigned to" is "Unassigned". There are also "Manage alert", "Export", and other buttons.

Figure 7.2 – Alert from MDI

For a specific alert, we can download a detailed report in Excel format for further analysis if needed. In this report, we will find a summary of the alert, the event activities that occurred, and the related entities.

When we click the **Manage alert** button, a new flyout menu appears so we can easily change the **Status**, **Assign to**, and **Classification** details.

Status can have one of the following values:

- New

- In Progress
- Resolved

Assign to is used to assign a specific alert or incident to a user or group within **Entra ID** (see *Figure 7.3*).

The screenshot shows the Microsoft Defender interface with a modal dialog titled "Manage incident". The main pane displays an "Attack story" for a "Suspected DCShadow attack (domain controller replication request)" on Jun 24, 2024 at 1:37 PM. The story includes tabs for "Alerts" (1), "Assets" (1), "Investigations" (0), and "Evidence and Response". The "Alerts" tab is selected, showing one new alert. The "Assign to" field in the dialog is populated with "SOC_L2_Analyst". Other fields include "Incident name" (Suspected DCShadow attack (domain controller replication request) on one end...), "Severity" (High), "Status" (Active), and "Classification" (Not set). Buttons for "Save" and "Cancel" are at the bottom right of the dialog.

Figure 7.3 – Manage incident

For the classification, we can choose one of the following sub-classifications:

- **True positive :**
 - Multi-staged attack
 - Malware
 - Malicious user activity
 - Unwanted software
 - Phishing
 - Compromised account
 - Other
- **Informational, expected activity (also known as Benign True Positives, or B-TPs for short) :**
 - Security testing
 - Confirmed activity

- Line of business application
- Other

- **False positive :**

- Not malicious
- Not enough data to validate
- Other

Having covered the choices for true positives, it's clear how these classifications play a critical role in identifying genuine threats. Now, let's delve into the **Informational, expected activity** classifications, which are also known as B-TPs. Understanding these will help us better manage and interpret alerts that, while initially flagged as potential threats, are ultimately determined to be legitimate and non-malicious activities.

The confusion matrix – understanding classification outcomes

When working with MDI and the other Defender products, it's crucial to understand the different types of outcomes that can result from security alerts. This is where the concept of the **confusion matrix** comes into play. The confusion matrix is a tool used to evaluate the performance of a classification system by breaking down its outcomes into four categories: **true positives**, **false positives**, **true negatives**, and **false negatives**. However, within the Microsoft Defender portal, we primarily use three classifications for alerts and incidents: **True positive**, **False positive**, and **Informational, expected activity**.

These outcomes help you evaluate the effectiveness of your detection systems and guide your response strategies. Let's break down all these terms in a way that's easy to grasp:

- **True positive (TP)** : Occurs when MDI correctly identifies a real threat. This means that the system's alert was accurate, and the detected activity is indeed malicious:
 - **Example** : MDI raises an alert for a suspicious login attempt, and upon investigation, you find that an unauthorized user was trying to access sensitive information. This alert is a TP because it correctly identified a genuine threat.
- **False positive (FP)** : Happens when MDI flags an activity as suspicious or malicious, but it's legitimate and harmless. While FPs can be frustrating, they are a common part of fine-tuning any security system:
 - **Example** : An alert is generated for a network scan, but it's later determined that the scan was conducted by your IT team as part of routine maintenance. This is a false positive because the alert was triggered by non-malicious activity.
- **False negative (FN)** : Happens when MDI fails to detect a real threat. This means a malicious activity occurred, but the system did not raise an alert. FNs are critical to identify and address because they represent missed threats:
 - **Example** : An attacker gains access to your network without triggering any alerts from MDI. This is a FN, indicating that the detection system did not catch the malicious activity.

- **True Negative (TN)** : Happens when MDI correctly identifies that there is no threat. This outcome is equally important as it confirms that normal, non-malicious activities are not being incorrectly flagged as suspicious:
 - **Example** : Regular user login activities that do not trigger any alerts. These activities are true negatives, showing that the system correctly identified them as benign.
- **Benign True Positive (B-TP)** : This is a specific type of TP where the system correctly identifies an activity that matches a threat pattern, but upon investigation, it turns out to be legitimate and non-malicious:
 - **Example** : MDI alerts you to an unusual login time for a user, but after checking, you find the user was working late on a critical project. This is a B-TP because the alert was accurate, but the activity was not harmful.
- **Informational alerts** : Provide details about expected and routine activities. These alerts are not indicative of any threat but are logged for visibility and auditing purposes:
 - **Example** : An alert indicating a successful daily backup process. This is expected behavior and is logged for record-keeping.

Understanding the difference between various classification outcomes is essential for improving your organization's security posture. By recognizing and addressing these outcomes, you can fine-tune your alert settings to reduce the number of FPs. This minimizes disruptions caused by non-threatening activities, allowing your security team to focus on genuine threats. You'll find more on how we can tune alerts in the next section, *Alert tuning*.

Addressing FNs is equally important. Enhancing your detection capabilities ensures that real threats are not overlooked. This means continuously improving your systems to catch malicious activities that might otherwise go unnoticed.

An efficient response strategy relies on the ability to prioritize TPs. When you correctly identify and swiftly respond to actual threats, you significantly reduce the potential damage to your organization. Prompt and accurate responses are crucial in maintaining a robust security defense.

Additionally, understanding B-TPs and informational alerts helps in maintaining contextual awareness. Recognizing when an alert signifies an expected, non-threatening activity prevents unnecessary investigations. This awareness not only streamlines your security operations but also ensures that your resources are used effectively.

To further understand these outcomes, it's helpful to explore the concept of the confusion matrix, which provides a detailed breakdown of classification outcomes:

	Actual Threat	No Threat
Predicted Threat	TP	FP
No Threat Predicted	FN	TN

To summarize, the confusion matrix provides a comprehensive framework for evaluating and improving the performance of your detection systems. By understanding and applying its principles, you can enhance your ability to identify real threats, reduce false alarms, and maintain a robust security posture.

Alert tuning

Effective alert tuning is essential for optimizing the performance of MDI and ensuring that your security team can focus on the most critical threats. This section will guide you through the process of tuning and suppressing alerts in Defender XDR, as well as adjusting alert thresholds in MDI. It's important to approach these configurations with caution, as improper settings can either overwhelm your team with too many alerts or, conversely, miss critical threats.

IMPORTANT

While tuning and adjusting thresholds, it's crucial to approach these configurations with caution. Over-tuning can lead to missing critical alerts, while under-tuning can overwhelm your team with too many alerts. Always test changes in a controlled environment before applying them broadly. Regularly review and adjust these settings to ensure they remain aligned with your evolving security needs.

Tuning and suppressing alerts in Defender XDR

Tuning alerts helps reduce noise and ensures that your security team is not overwhelmed by non-critical alerts. By refining the alert criteria and suppressing unnecessary alerts, you can improve the efficiency of your incident response:

1. **Accessing the portal** : Access the Defender XDR portal by navigating to <https://security.microsoft.com/> with the appropriate permissions, such as Security Administrator or Security Operator.
2. **Navigating to Alert tuning** :
 - In the portal, navigate to **System > Settings** .
 - Select **Microsoft Defender XDR** .
 - Select **Alert tuning** under the **Rules** heading.
3. **Creating or editing rules** : To create a new tuning rule, select **Add new rule** . To edit an existing rule, click on the rule title to open its details page. Here, you can view associated alerts, edit conditions, or toggle the rule on and off.
4. **Configuring service sources** : In the **Tune alert** pane, select the service sources where you want the rule to apply. Only services where you have permissions will be listed. For MDI, select **Microsoft Defender for Identity** .
5. **Adding conditions** :
 - Define conditions for alert suppression. For instance, to prevent an alert when a trusted internal scanner IP is detected, set a condition for the IP **Custom trigger** , and specify the IP details. Conditions can be based on various **indicators of compromise** (IOCs), such as IP addresses, processes, and Active Directory changes.

- Use **AND**, **OR**, and grouping options to define the relationships between multiple evidence types. For example, you might create a condition that suppresses alerts only if both a scheduled maintenance process and a specific file change occur within the same timeframe. Another example could be suppressing alerts for known benign activities in Active Directory, such as automated user account creation by a trusted provisioning system.

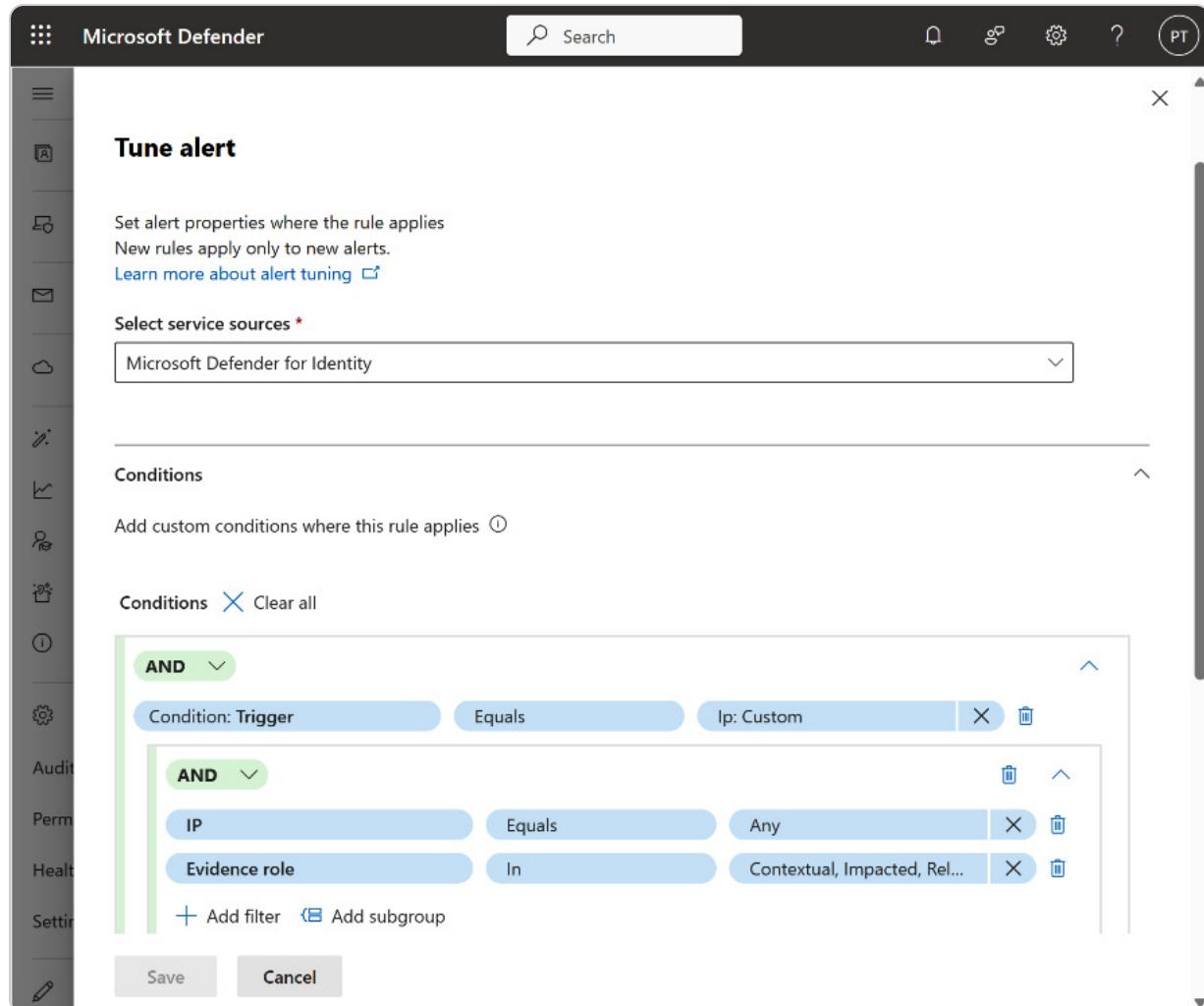


Figure 7.4 – Alert tuning in Defender

6. **Selecting actions** : When you are done with the conditions, choose the action you want the rule to take – either **Hide alert** or **Resolve alert**.
7. **Saving the rule** : Enter a meaningful name and comment for the rule, then save your changes.

The importance of re-evaluating alert tuning rules

It's essential to regularly review and re-evaluate your alert tuning rules. This ensures that suppressed alerts are still relevant and that no critical or vital alerts are missed. Regular re-evaluation helps maintain an optimal balance between reducing noise and ensuring important threats are detected. This includes two steps:

- **Revisit tuning rules** : Periodically review your alert tuning rules to ensure they remain effective. As your environment changes, previously benign activities might become suspicious, and new benign activities might need suppression.
- **Monitor suppressed alerts** : Regularly check the alerts that have been hidden or resolved due to tuning rules. Ensure that those alerts are not overlooked by the SecOps team, adjusting your rules as necessary based on these reviews.

Adjusting alert thresholds in MDI

Adjusting alert thresholds in MDI is a crucial task that can significantly impact your security monitoring and response strategy. Before making any changes, it's important to understand the implications and how to manage these settings effectively.

THE IMPORTANCE OF KNOWING BEFORE CHANGING SETTINGS

Before adjusting the alert thresholds, it's vital to grasp how these settings influence the detection and reporting of suspicious activities. Modifying thresholds can affect the volume of alerts you receive and their relevance, which directly impacts your security team's workload and the organization's overall security posture. Misconfigurations can either flood your team with too many alerts, including FPs, or miss critical threats by being too restrictive.

By default, MDI alerts are set to a **High** threshold, which balances sensitivity to detect genuine threats while minimizing FPs. The **Recommended Test Mode** is a special configuration that sets all alert thresholds to **Low**. This mode is useful for testing and evaluation purposes, as it generates a higher volume of alerts, including those related to legitimate activities. Using Test Mode helps in identifying and fine-tuning alert settings before applying them broadly.

The threshold levels in MDI can be set to **High**, **Medium**, or **Low**. Here's what each level means:

- **High (default)** : This is the standard setting that aims to detect significant threats while keeping FPs to a minimum. It ensures that only the most critical and suspicious activities trigger alerts.
- **Medium** : This setting increases the sensitivity of alerts, capturing more activities that might be indicative of a threat. While it detects a broader range of suspicious behaviors, it may also increase the number of FPs.
- **Low** : This setting maximizes sensitivity, generating a high volume of alerts for a wide range of activities, including many that are benign. This level is typically used in Test Mode to ensure comprehensive detection during evaluation and fine-tuning.

The following steps have to be followed to adjust alert thresholds:

1. **Accessing the portal :**

- Access the Defender XDR portal by navigating to <https://security.microsoft.com/> with the appropriate permissions, such as **Security Administrator**, **Security Operator**, or a **Unified RBAC role**.

2. **Navigating to the MDI settings :**

- In the portal, navigate to **System > Settings** .
- Select **Identities** .

3. Access the **Adjust alerts thresholds** menu. Under the **General** heading, select **Adjust alerts thresholds** .

4. Modify **Threshold levels** or enable **Recommended Test Mode** (if needed):

- **Test Mode** : To thoroughly evaluate the impact of threshold changes, enable **Test Mode** . This will set all alert thresholds to **Low** , generating a higher volume of alerts for wide-ranging testing.
- **Alerts** : In the table right under the **Test Mode** toggle, you can choose the specific alert you want to adjust. Select the desired threshold level (**High** , **Medium** , or **Low**) based on your organization's needs for each of the alerts.

Please take note of the Information column which describes the specific conditions under which each alert is triggered and how these conditions change based on the threshold level. For example, the **Security principal reconnaissance (LDAP)** alert on **Medium** mode triggers immediately and disables filtering of popular queries, while on **Low** mode, it includes all **Medium** mode criteria plus a lower threshold for queries and single-scope enumeration.

Similarly, the **Suspected Brute Force attack (Kerberos, NTLM)** alert on **Medium** mode ignores previous learning with a lower threshold for failed passwords, and on **Low** mode, it sets the lowest possible threshold. Each alert type will have detailed descriptions about the varying levels of sensitivity and triggering conditions to help you understand what to expect and how to configure them effectively.

5. **Save changes** :

After making the necessary adjustments, click **Apply changes** to save the new settings.

Regularly reviewing and adjusting these thresholds ensures that your alerting system stays aligned with your security requirements and adapts to any changes in your environment. We will now look at the **User Entity** page within Defender XDR to get all types of information that we have on our users.

User Entity

The **User Entity** page in the Microsoft Defender XDR portal is a crucial tool for investigating user accounts involved in alerts or incidents. It consolidates all the relevant details about a particular user, allowing you to assess any potential security risks they pose. If a user is flagged in an alert or incident, this page becomes a key part of your investigation workflow.

You can access detailed user information from several locations within the Defender XDR portal:

- **Identities** page, under **Assets**
- Alerts queue
- Any individual alert/incident
- **Devices** page
- Any individual device entity page
- Activity log

- Advanced hunting queries
- Action center

The Overview tab

The **Overview** tab contains the following information:

- **Entity Details** : The **Entity Details** panel on the left side of the **Overview** page gives you critical information about the user immediately. You'll find the Microsoft Entra identity risk level, details on the devices the user has signed in to, timestamps of when the user was first and last seen, user accounts, group memberships, contact information, and more. Depending on your integration settings, additional details may be visible, offering a deeper view of the user's identity profile.
- **Incidents and Alerts** : This card provides a visual summary of all incidents and alerts linked to the user. The alerts are grouped by severity, helping you quickly assess any current or past security issues involving this user entity.
- **Investigation Priority** : The **Investigation Priority** card shows the calculated priority score for the user, along with a two-week trend for that score. This view allows you to track how the user entity's risk profile has evolved over time, as well as compare it to other users in your tenant, thanks to percentile-based scoring.
- **Active Directory Account Controls** : Here, you'll find security settings from MDI that require attention. Flags such as whether the user can bypass a password with the *Enter* key or if the user's password never expires are listed here, providing insights into potential weaknesses in the account's configuration.
- **Scored Activities** : The **Scored Activities** card captures all activities and alerts that have contributed to the user's investigation priority score in the past seven days. This gives a clear overview of why the user might be deemed a security risk.
- **Organization Tree** : This section provides a hierarchical view of where the user fits within your organization, as reported by MDI. This is useful when assessing the user's access level and their potential role within the organization.
- **Account Tags** : Tags pulled from Active Directory offer an additional layer of context for user monitoring. These tags include the following:
 - **New** : The account was created less than 30 days ago.
 - **Deleted** : The account has been permanently removed from Active Directory.
 - **Disabled** : The account is currently inactive and cannot be used for sign-in.
 - **Enabled** : The account is active and can be used within the domain.
 - **Expired** : The account's expiry date has passed, making it unusable for sign-in or resource access.
 - **Honeytoken** : A manually tagged decoy account set up to detect unauthorized access.
 - **Locked** : The account is locked due to multiple failed password attempts.
 - **Partial** : The account is not fully synchronized with Active Directory, and some attributes may be missing.
 - **Unresolved** : The account does not correspond to a valid entity within the Active Directory forest.
 - **Sensitive** : The account is flagged as sensitive, requiring special monitoring or protection.

These tags are invaluable for understanding the current state and risks associated with each user account, allowing you to take the appropriate action based on the status of the account.

The Incidents and Alerts tab

The **Incidents and Alerts** tab offers a detailed view of all active incidents and alerts involving the selected user, going back up to six months. Here, you'll see all the relevant information pulled directly from the main incidents and alerts queues, but filtered specifically for this user. For each entry, a short description is provided along with key details, such as the following:

- **Severity** : Whether the alert is classified as **High**, **Medium**, **Low**, or **Informational**
- **Status** : The current state of the incident or alert, such as new, in progress, or resolved
- **Classification** : Whether the alert has been marked as not set, a false alert, or a true alert
- **Investigation state** : The current phase of the investigation
- **Category** : The type of issue identified
- **Assigned Analyst** : The person responsible for addressing the alert
- **Last Activity** : The most recent event associated with the incident or alert

This tab provides a streamlined way to review all potential security incidents associated with the user, offering insights into the user's activity and highlighting any areas of concern.

You can customize the view to display a different number of entries per page (the default is 30) and adjust which columns are shown. Additionally, filtering options allow you to narrow down incidents and alerts by severity, status, or any other column.

The **Impacted Entities** column displays all the devices and user entities referenced in each alert or incident, making it easier to assess the broader impact of any given event.

When you select an incident or alert, a fly-out panel appears on the side of the screen, where you can view more in-depth details, such as the incident or alert number, related devices, and more. This panel also allows you to manage the selected incident or alert directly from the fly-out. You can also select multiple alerts at once to view or manage them in bulk.

The Observed in Organization tab

The **Observed in Organization** tab provides a comprehensive view of the user's interactions within your environment. It surfaces valuable information from various sources to help security teams better understand the user's behavior and detect any potential anomalies:

- **Devices** : This section lists all devices the user has signed into over the past 180 days, highlighting both frequently and infrequently accessed devices. This insight helps you identify whether a user's activity on certain devices is typical or warrants further investigation.
- **Locations** : Here, you'll find all the observed geographical locations for the user in the last 30 days. By monitoring location data, you can quickly spot any suspicious logins from unexpected regions, which may indicate a compromised account.
- **Groups** : This section outlines the on-premises groups the user is a member of, as reported by Microsoft Defender for Identity. It gives you a clear understanding of the user's group memberships, which is crucial for evaluating their permissions and potential risks.

- **Lateral Movement Paths** : MDI's detection engine identifies potential lateral movement paths, showing how an attacker might move through your network using this user's account. These paths give visibility into vulnerabilities that attackers could exploit to escalate privileges or move deeper into your network.

The Timeline tab

The **Timeline** tab provides a consolidated view of all user activities and alerts from the past 180 days. It brings together entries from multiple Defender workloads, including Microsoft Defender for Identity, Microsoft Defender for Cloud Apps, and Microsoft Defender for Endpoint. This unified view allows you to easily track a user's activity across different services. By using the timeline, you can filter and focus on specific actions a user performed or actions taken against their identity, enabling more targeted investigations based on a selected timeframe.

Time	Activity	Type	Device / IP A...	Alert severity	Status
Oct 18, 2024 8:55:27 AM	Account enumeration reconnaissance	Alert	B_05	Medium	New
Oct 18, 2024 8:36:38 AM	Account enumeration reconnaissance	Alert	B_06	Medium	New
Oct 18, 2024 4:12:45 AM	Account enumeration reconnaissance	Alert	SERVER	Medium	New

Figure 7.5 – User timeline

By using the timeline, you can filter and focus on specific actions a user performed or actions taken against their identity, enabling more targeted investigations based on a selected timeframe.

The Policies tab

In the **Policies** tab, we will find any Conditional Access policy that is applied to the specific user. We can access the policies in the new Microsoft Entra portal (entra.microsoft.com) by clicking on the policy name.

Name	Active	Target Apps	Grant Controls
All Apps: Block Legacy Auth	On	All	block
All Apps: Require MFA For all Users Whe...	On	All	mfa
Office 365: Require approved client apps ...	On	1 App	approvedApplication
Trigger MFA request when High & Medi...	On	All	mfa

Figure 7.6 – Conditional Access Policies

We will now look at one of the tabs in more detail that we need to take seriously, and that's the lateral movement path.

Lateral movement paths (LMPs)

Lateral movement is a technique where attackers move through a network by compromising non-sensitive accounts to eventually access high-value, sensitive accounts. This method allows attackers to exploit shared credentials and gain control of critical systems such as domain controllers. MDI actively monitors for lateral movement threats using its **Security Alerts** feature to detect these types of activities.

One of the standout features of MDI is **Lateral Movement Paths (LMPs)**. LMPs visually show how an attacker could move from one account or machine to another, helping SecOps easily spot vulnerable areas within the network. By mapping these potential pathways, LMPs help identify and block the routes attackers could take to compromise sensitive accounts.

Some of the most common lateral movement techniques include credential theft and pass-the-ticket attacks, where attackers use compromised non-sensitive accounts to move through machines that

store valuable credentials. These attacks often exploit weak points in the sharing of credentials between accounts and systems.

MDI's LMPs provide insight based on the type of entity:

- **Sensitive users** : LMPs show potential paths leading to the user
- **Non-sensitive users or computers** : LMPs show how they may be part of an attacker's lateral movement

Each LMP is saved for 48 hours, allowing past movements to be reviewed, assisting SecOps in investigating potential attacks over time. The ability to analyze which non-sensitive user initiated the lateral movement offers further clarity on how a threat could unfold within the network, providing a clearer view of an attack's starting point and progress.

To begin investigating LMPs, you will start by exploring the Microsoft Defender XDR portal. From there, you can search for specific users or entities that may be part of an LMP. The process allows you to track and assess how attackers could potentially exploit weaknesses, use stored credentials, and move closer to your sensitive accounts. The following steps will guide you through this investigation process:

1. Navigate to <https://security.microsoft.com> with the appropriate permissions, such as **Security Administrator**, **Security Operator**, or a **Unified RBAC** role.
2. While logged in, search for the desired user to bring up their profile.
3. In the user profile, under the **Observed in Organization** section (available in both the **Overview** and **Observed** tabs), check whether the user is identified in any potential LMP.
4. If the user is flagged in an LMP, click the **Lateral Movement Paths** option under the **Observed in Organization** tab.
5. The system will display a graph illustrating the possible paths from low-privileged accounts or devices to the sensitive user account during the last 48 hours.
6. Review the graph to understand how attackers could potentially exploit access to reach a sensitive user.
7. Use the **Select a Date** option to view the graph for LMP detections at different times, allowing you to analyze previous exposures and patterns.

For example, the graph might show arrows such as **Logged into by**, which would detail which users have logged in to certain devices. This could reveal whether a high-privileged user's credentials were saved on a vulnerable machine. Further, observe which other users have access to the same machines, highlighting any increased exposure or risk of credential compromise.

You can also use **Advanced Hunting** queries to identify accounts that might be part of a potential lateral movement path. This allows you to proactively track suspicious activity across your network and focus on key accounts that are at risk. The following query will help identify events where a potential lateral movement path has been flagged:

```
IdentityDirectoryEvents  
| where ActionType == "Potential lateral movement path identified"
```

This query filters the events to focus on those specifically labeled as lateral movement paths, giving you a starting point for further investigation:

The screenshot shows the Microsoft Defender Advanced Hunting interface. At the top, there are buttons for 'Run query', 'Last 30 days', 'Save', 'Share link', and 'Manage rules'. Below this is a 'Query' section with the following code:

```
1 IdentityDirectoryEvents
2 | where ActionType == "Potential lateral movement path identified"
```

Below the query, there are tabs for 'Getting started', 'Results' (which is selected), and 'Query history'. Under 'Results', there are buttons for 'Export', 'Show empty columns', and a search bar. It shows 1 item found in 00:00:135, with a 'Low' severity indicator. There are also icons for filtering and sorting.

Under 'Filters', there are dropdowns for 'TimeGenerated', 'Timestamp', 'ActionType', and 'Application'. The 'ActionType' dropdown is set to 'Potential lateral movement path identified' and 'Application' is set to 'Active Directory'.

Figure 7.7 – Advanced Hunting for LMPs

Make sure to regularly review the **Riskiest Lateral Movement Paths (LMP)** security assessment under **Secure Score recommendations**. As highlighted in [Chapter 10](#) and within the *Operational Guide*, reviewing Secure Score recommendations weekly is highly recommended to stay ahead of potential risks.

While LMPs are helpful in investigating and mitigating ongoing threats, they also play a vital role in preventing future attacks. By visualizing potential attack paths, you can take proactive steps to eliminate these vulnerabilities before attackers gain control of sensitive accounts. One of the best practices for reducing lateral movement is the Active Directory tiering model, which limits the exposure of high-privileged credentials to lower-tier systems. For example, ensuring that Tier 0 credentials (such as domain admins) are never used on lower-tier devices or accounts helps prevent attackers from gaining domain dominance via lateral movement.

To further reduce lateral movement risks, ensure that sensitive users log in to hardened devices using their admin credentials only when necessary. For example, if an administrator is flagged in an LMP, assess whether they need access to a shared device. If so, enforce the use of non-admin credentials for daily activities and reserve privileged accounts for high-security tasks. Additionally, review group memberships regularly to ensure that only essential users retain administrative rights.

We will now look at how we can triage our alerts and incidents.

Initial triage and categorization

Effective initial triage and categorization of alerts are critical steps in managing security incidents within all **Microsoft Defender** products. This process helps in quickly identifying, prioritizing, and

responding to potential threats, ensuring that resources are efficiently allocated, and critical threats are addressed promptly.

NOTE – UNDERSTANDING ALERTS VERSUS INCIDENTS

Alert : A notification for a single potential security threat or suspicious activity detected by Defender products.

Incident : A collection of related alerts grouped together to represent a broader security event or attack.

Alerts provide specific details about individual events, while incidents give a comprehensive view by aggregating related alerts.

When an alert or incident is raised in the **Microsoft Defender XDR portal**, the new unified portal, they are correlated with information and data from other Microsoft Defender products. This correlation provides a comprehensive view of the security landscape, enabling more accurate and efficient threat management.

But before we jump into the correlation and how we progress in the alert investigation, here's how to approach the initial triage and categorization.

Steps to take immediately after receiving an alert or incident

When an alert and incident is received, the first step is to quickly assess its validity and potential impact. This involves examining other detection sources (such as Defender for Endpoint or other Defender products) and the details provided by MDI, such as the nature of the activity, the entities involved, and the context of the alert/incident.

Before diving into the immediate actions, we need to access the **Defender XDR portal** and locate the alerts and incident section. Follow these steps:

1. Navigate to <https://security.microsoft.com> with the appropriate permissions, such as **Security Administrator**, **Security Operator**, or a **Unified RBAC** role.
2. Click on **Investigation & response**.
3. Under **Incident & alerts**, choose one of the following:
 - For alerts, click on **Alerts**.
 - For incidents, click on **Incidents**.

Once you have located the alerts, the following immediate actions should be taken.

1. **Review details** : Analyze the information provided by the alert, including the type of threat, affected entities and the count of those entities, and any associated activities, and if possible, understand the rule behind the alert (if an analytic rule in **Microsoft Sentinel**, or a Custom Detection in Defender XDR, see the **KQL query**; if an **MDI alert**, check the **official documentation** about the alert).
2. **Check for known issues** : Cross-reference the alert with known issues or ongoing investigations to determine if it is part of a larger pattern.

3. **Gather additional context** : Use MDI's features to gather more context, such as looking at the activity timeline, related alerts, and entity tags.

4. **Enrichments** : Get more information about external/source IP addresses and how long have those IP addresses been in the environment, sign-ins for the user, the location of the sign-ins, see any suspicious pattern for the user, see whether the same file or file hash appears on other endpoints, and so on. Here are some external resources to utilize for your enrichment:

- **AbuseIPDB** : A database of IP addresses linked to malicious activities. Useful for checking whether an IP address associated with an alert has been reported for abusive behavior.
- **VirusTotal** : An online service that analyzes files and URLs for viruses, worms, trojans, and other kinds of malicious content. It can provide detailed reports on the threat level of a file or URL.
- **Have I Been Pwned** : A resource for checking whether an email address or domain has been part of a data breach, useful for understanding the potential exposure of user credentials.
- **Shodan** : A search engine for internet-connected devices. It can be used to find out more about the devices and services exposed by an IP address.
- **Cymon** : A threat intelligence aggregator that provides information on malicious IPs, domains, and URLs. It can help determine the reputation of an IP address or domain.
- **IPinfo.io** : Provides detailed information about IP addresses, including the geographic location, ISP, and type of IP (e.g., residential, business, hosting).
- **Cisco Talos Intelligence** : Offers comprehensive threat intelligence reports and analysis, useful for understanding the broader context of a threat.
- **AlienVault Open Threat Exchange (OTX)** : A collaborative platform where security professionals share threat data. It can help in identifying new threats and their IOCs.
- **Microsoft Defender Threat Intelligence (MDTI)** : If licensed for MDTI Premium or using **Microsoft Copilot for Security**, you can leverage datasets for detailed threat intelligence analysis. MDTI includes advanced datasets such as DNS, WHOIS, PDNS, trackers, host pairs, and cookies, providing context on internet infrastructure and detecting threat actor activity. For example, MDTI can reveal relationships between IP addresses and domains, shared TLS certificates, or host pairs, which help map potential threat pathways and connections.

NOTE

If you find that some actions need to be taken quickly to reduce more impact of an ongoing attack, like isolating devices, disabling accounts, and blocking IP addresses, please do so.

5. Responding with action, ensure that the MDI action account is configured correctly for reliable response actions, as detailed in [Chapter 8](#):

- **Disable user** : When selecting the **Disable user** option, a prompt will request confirmation. Upon confirmation, the system shows a status indicating that the action is in progress; however, this is not a guarantee that the action was successfully applied in **Active Directory (AD)**. You can verify the result in the Action Center history, where each response is marked as **Succeeded** or **Failure** :
 - **Validation** : To confirm the action was applied in AD, look for Event IDs 4725 (account disabled) and 4738 (attribute change). Use **Advanced Hunting** within Defender XDR to monitor MDI actions to validate outcomes. You can look in the **IdentityDirectoryEvents** table but also the **SecurityEvent** table (Microsoft Sentinel) if you have that configured:

```
IdentityDirectoryEvents  
| where ActionType == "Account disabled"
```

- **Force Password Reset** : This option triggers a prompt for the user to change their password at the next logon. However, note that this action is limited to AD and hybrid users. If the **Password never expires** setting is enabled, the system will show the action as **Successful** in Action Center but will not enforce a password reset:
 - **Validation** : Similar to disabling a user, you can verify the password reset through the event ID 4738, which logs modifications to account attributes.

Microsoft assists in generating incident names by utilizing various alert attributes such as detection sources, the number of endpoints involved, and other relevant categories. Additionally, if you have analytic rules configured within **Microsoft Sentinel** that generate incidents, those incident names may also be automatically adjusted based on these attributes. This dynamic naming helps provide a clearer context and more precise identification of incidents.

See examples of incident names here:

- *Suspected DCShadow attack (domain controller promotion) on one endpoint*

Explanation: This alert indicates that a machine in the network is attempting to register itself as a rogue domain controller, potentially using DCShadow techniques to replicate unauthorized changes to Active Directory objects through RPC and LDAP.

- *Lateral movement using remote logon by contained user blocked on multiple devices*

Explanation: Indicates that the **Attack Disruption** feature has been blocking the user account for further lateral movement. In this case, it is important to investigate the full attack.

- *Multi-stage incident involving initial access and execution on one endpoint reported by multiple sources*

Explanation: Indicates a complex attack that has been detected through various stages and verified by different security signals or products within the XDR suite.

Categorizing alerts by severity and potential impact

Once the initial assessment is done, the next step is to categorize the alert and incident based on their severity and potential impact. This categorization helps prioritize the response effort. The common categories include the following:

- **Informational** : Alerts that indicate expected or routine activity. These do not require immediate action but should be logged for visibility and auditing purposes:
 - **Example:** Scheduled system backups or routine network scans.
- **Low** : Alerts that represent minor security issues or anomalies. These might require monitoring or minor adjustments but are not immediate threats:
 - **Example:** Unusual login times from a known employee with a history of such behavior.

- **Medium** : Alerts indicating potential security threats that require further investigation and a prompt response. These could indicate the early stages of a larger attack:
 - **Example:** Multiple failed login attempts followed by a successful login from an unusual location.
- **High** : Alerts that represent serious security threats needing immediate action. These usually indicate ongoing or imminent attacks that could have significant consequences:
 - **Example:** Detection of malware on a critical server or unauthorized access to sensitive data.

Prioritizing alerts based on criticality and context

After categorizing alerts by severity, it is important to prioritize them based on the criticality of the affected systems and the context of the activity. This involves the following:

- **Assessing the impact** : Determine the potential impact of the alert on business operations, data integrity, and overall security posture.
- **Evaluating the context and the additional enrichment** : Consider the context in which the alert occurred, such as recent changes in the network, ongoing projects, or specific threats targeting your industry.
- **Assigning priority levels** : Assign priority levels to alerts to help manage response efforts. High-priority alerts should be addressed first to mitigate the most significant risks.

Implementing automated triage processes

To enhance the efficiency of triage and categorization, consider implementing automated processes using **Security Orchestration, Automation, and Response (SOAR)** tools. Automation can significantly streamline your security operations, enabling faster and more consistent responses to threats. Here are some of the benefits of automation in the triage phase:

- **Streamline initial assessment** : Automate the collection and analysis of alert details, reducing the time needed for initial triage. Automated systems can quickly gather relevant information such as IP addresses, user activity, and affected systems, allowing your security team to make informed decisions more rapidly.
- **Standardize categorization** : Use predefined rules to consistently categorize alerts based on severity and context. This ensures that alerts are classified uniformly, making it easier to prioritize and address them appropriately.
- **Gather more context and enrichment** : Automate the process of enriching alerts with additional context, such as threat intelligence data, historical incident data, and related alerts. This provides a more comprehensive view of the potential threat, aiding in quicker and more accurate decision-making.
- **Accelerate response** : Automatically prioritize alerts and trigger response workflows for high-severity incidents, ensuring quick and effective action. Automation can initiate immediate responses, such as isolating affected systems, notifying relevant stakeholders, and beginning the investigation process.

Once we have done the initial triage and categorization of the alert and incident, the next step is to understand *why* this happened in the first place.

Root cause analysis

Root Cause Analysis (RCA) is a crucial aspect of incident response and investigation within cybersecurity. It involves identifying the underlying causes of security incidents to prevent their recurrence. In the context of MDI, RCA helps security teams understand how an incident occurred, its impact, and the steps necessary to mitigate similar future incidents. The following steps are involved in RCA:

1. Data collection :

- **Gather all relevant data related to the incident**, including logs, alerts, and any other pertinent information from **MDI**, **Microsoft Sentinel**, and other integrated security tools.
- **Utilize KQL queries** to extract detailed logs and events that can shed light on the incident's timeline and activities.

2. Incident timeline reconstruction :

- **Construct a detailed timeline** of the incident, outlining the sequence of events from the initial detection to the final resolution.
- **Identify key events** that contributed to the incident, such as unauthorized access, suspicious activities, and system changes.

3. Identification of root causes :

- **Analyze the collected data** to determine the primary factors that led to the incident. This could include vulnerabilities, misconfigurations, user errors, or **advanced persistent threats (APTs)**.
- **Use tools** like **Microsoft Defender** and **Microsoft Sentinel** to correlate events and identify patterns that may indicate the root cause.

4. Impact assessment :

- **Evaluate the impact** of the incident on the organization. This includes assessing the extent of data exposure, compromised systems, and potential business disruptions.
- **Use impact assessment** to prioritize remediation efforts and allocate resources effectively.

5. Mitigation and remediation :

- **Develop a remediation plan** to address the root causes and prevent future incidents. This may involve patching vulnerabilities, updating security policies, and enhancing monitoring capabilities.
- **Implement automated response actions** using **SOAR** tools to streamline remediation processes and ensure timely intervention.

6. Documentation and reporting :

- **Document the findings** of the RCA, including the identified root causes, impact assessment, and remediation steps.
- **Prepare comprehensive reports** for stakeholders, detailing the incident and the actions taken to resolve it.

7. Continuous improvement :

- **Review the RCA process and outcomes** to identify areas for improvement in the incident response strategy.

- Update security protocols and training programs based on the lessons learned from the incident.

RCA is crucial for enhancing the security posture of an organization, particularly following a security breach. By identifying and understanding the underlying causes of incidents, security teams can effectively mitigate vulnerabilities. Addressing these root causes allows for the resolution of issues that contributed to the incident and helps prevent future occurrences by learning from past events.

Furthermore, RCA provides valuable insights that refine and improve incident response strategies, making them more efficient and effective. This systematic approach ensures that security measures are continuously optimized to respond to emerging threats and evolving security landscapes.

By leveraging tools and techniques such as KQL for querying, analyzing data, and identifying patterns, security teams can utilize SOAR tools to automate the RCA processes. This leads to a better understanding of the environment, enabling more proactive and strategic responses to potential threats.

Real-world playbook – responding to advanced threats

In this section, we delve into a comprehensive playbook designed to guide security teams through the detection, validation, and response to advanced threats using MDI, **Microsoft Sentinel**, and other integrated security tools. This structured approach leverages automation, SOAR capabilities, and threat intelligence to manage complex security incidents effectively.

Defining advanced threats

“The pace of change is never going to be this slow again.”

This quote has rung true since I first heard it back in 2020 from an old colleague. In the fast-evolving landscape of hyperscalers, AI, and cybersecurity, advanced threats represent a significant challenge to organizations. Understanding these threats is essential to stay ahead of the curve to protect digital assets effectively.

Advanced threats in cybersecurity refer to sophisticated, often targeted attacks designed to infiltrate, disrupt, or exploit an organization's digital infrastructure. Unlike basic cyber threats, which may rely on well-known vulnerabilities and techniques, advanced threats are characterized by their complexity, persistence, and the significant damage they can inflict. These threats often come from highly skilled attackers, including nation-states, organized crime groups, and professional cybercriminals.

One of the primary characteristics of advanced threats is their sophistication. Attackers use **cutting-edge technology**, **custom malware**, and exploit **zero-day vulnerabilities** – flaws that are unknown to the software vendor and the general public. These sophisticated attacks require extensive knowledge and resources, making them particularly dangerous.

Persistence is another hallmark of advanced threats. Attackers are patient and strategic, often spending months or even years conducting reconnaissance, establishing footholds, and moving laterally within a network without detection. This persistence allows them to gather valuable information and plan their attacks meticulously, increasing the potential for damage.

Common techniques used by attackers include lateral movement and privilege escalation. Once inside a network, attackers move laterally to access additional systems and data. They often use legitimate tools such as **PowerShell** and RDP to navigate the network stealthily. Attackers seek to gain higher-level permissions to access sensitive data and critical systems. This can be achieved through exploiting software vulnerabilities, credential theft, or leveraging misconfigurations.

Advanced threats are also highly targeted. Unlike opportunistic attacks, which cast a wide net in hopes of catching vulnerable victims, advanced threats focus on specific organizations or sectors with valuable data or critical operations. This targeting is often based on extensive research and planning, making the attacks more effective and harder to defend against.

Furthermore, advanced threats typically employ multiple attack vectors. Attackers might use a combination of phishing, malware, ransomware, and insider threats to achieve their objectives. This multi-vector approach complicates detection and response efforts, as defenders must monitor and protect against a broader range of potential entry points and attack methods.

One projecting example of an advanced threat is APTs. APTs are prolonged and targeted attacks aimed at stealing sensitive information or compromising critical infrastructure. These attacks often

involve multiple stages, from initial intrusion to data exfiltration, and are carried out over extended periods.

Human-operated ransomware represents another significant advanced threat. Unlike automated ransomware attacks, which use pre-programmed scripts to spread malware, human-operated ransomware involves skilled attackers manually deploying ransomware after gaining extensive access to an organization's network. This type of attack often begins with compromised credentials or exploiting vulnerabilities, allowing the attackers to navigate the network and identify valuable targets before encrypting data and demanding a ransom.

Supply chain attacks are also increasingly common and concerning. In these attacks, cybercriminals compromise software or hardware from trusted vendors, embedding malicious code that can spread to the vendor's customers. This type of attack exploits the trust relationships between vendors and their clients, making it particularly insidious and difficult to detect.

To combat these sophisticated threats, organizations must employ advanced threat detection and response strategies. These strategies involve a multi-layered approach, integrating technology, threat intelligence, and human expertise to identify and mitigate cyber threats promptly.

The **People, Process, Technology** (PPT) framework is a holistic approach used to enhance organizational efficiency and effectiveness. By integrating these three critical components, organizations can achieve their strategic goals and maintain a competitive edge. Here's a brief overview of each element in the context of cybersecurity:

- **People :**

- Skilled cybersecurity professionals
- Continuous training and awareness programs
- Clear roles and responsibilities

- **Process :**

- Well-defined incident response plan
- Regular security audits and assessments
- **Standard Operating Procedures** (SOPs) for threat detection and response

- **Technology :**

- Advanced threat detection tools (e.g., AI, anomaly detection)
- **Security Information and Event Management** (SIEM) systems
- Automated response and containment solutions

Detection is the first critical step in the process of battling cyber threats. Organizations must continuously monitor their endpoints, networks, and cloud environments for unusual activity and known threat signatures. Advanced detection methods are essential in this effort. Behavior-based detection focuses on identifying abnormal behaviors that may indicate a compromise, while anomaly detection uses AI and analytics to recognize deviations from normal activity patterns. Additionally, threat hunting involves proactively searching for indicators of compromise that may not trigger automated alerts, allowing security teams to uncover and address potential threats before they cause significant harm.

Once a threat is detected, a **rapid and coordinated response** is crucial. Effective response strategies include containment, which involves isolating affected systems to prevent further spread of the threat. Eradication follows, aiming to remove the threat from the environment, including the root cause and any lingering components. Recovery efforts then restore normal operations and strengthen defenses to prevent future incidents. Reporting and mitigation complete the process, as security teams document the incident, assess its impact, and implement measures to enhance the organization's security posture.

Advanced threats represent a significant challenge to cybersecurity, requiring organizations to adopt **robust and adaptive defense mechanisms**. By understanding the nature of these threats and implementing comprehensive detection and response strategies, organizations can better protect their assets and maintain resilience in the face of an ever-changing threat landscape. The evolving nature of cyber threats demands vigilance, advanced technology, and a proactive approach to cybersecurity to ensure that organizations remain secure and resilient against the most sophisticated and persistent attacks.

Identifying the characteristics of an advanced threat involves understanding the sophisticated methods and strategic approaches that differentiate these threats from more basic cyber threats. The complexity and persistence of advanced threats make them particularly challenging to detect and mitigate. Here are the key characteristics and the tactics used by attackers:

- **Sophistication** : Advanced threats often utilize cutting-edge technology, custom malware, and exploit zero-day vulnerabilities – flaws that are unknown to both the software vendor and the public. Attackers with extensive technical knowledge and resources create sophisticated attacks that are difficult to detect and mitigate.
- **Persistence** : Attackers behind advanced threats are patient and methodical. They may spend months or even years conducting reconnaissance, establishing footholds within a network, and moving laterally without detection. This persistence allows them to meticulously gather valuable information and plan their attacks, increasing the potential for significant damage.
- **Targeted nature** : Advanced threats are highly targeted, focusing on specific organizations or sectors with valuable data or critical operations. This targeting is usually based on thorough research and planning, making the attacks more effective and challenging to defend against.
- **Multi-vector approach** : Advanced threats frequently employ multiple attack vectors to achieve their objectives. This might include a combination of phishing, malware, ransomware, and insider threats. Such a multi-vector approach complicates detection and response efforts, as defenders must protect against a broader range of potential entry points and attack methods.

- **High impact** : The potential damage from advanced threats is significant, often involving data breaches, operational disruptions, or intellectual property theft. The impact is amplified by the attackers' strategic planning and extensive access gained through sophisticated techniques.

As we move forward, we'll explore examples of advanced threat tactics, such as phishing, malware, and ransomware. These examples will show how attackers use different methods to infiltrate and compromise systems.

Examples of advanced threat tactics

- **Phishing campaigns** : One of the most common methods bad actors use to infiltrate a company is through phishing campaigns. These attacks involve sending emails that trick employees into downloading malicious code or providing their credentials.
- **Malware** : Cyber attackers deploy software designed to damage computers and systems or collect sensitive information. Malware can range from viruses and worms to sophisticated spyware and trojans.
- **Ransomware** : A particularly destructive type of malware, ransomware attackers hold critical systems and data hostage, threatening to release private data or steal cloud resources to mine Bitcoin until a ransom is paid. Human-operated ransomware, in which a group of cyber attackers gains access to an organization's entire network, has become a growing issue for security teams.
- **Distributed Denial-of-Service (DDoS) attacks** : Using a series of bots, bad actors disrupt a website or service by flooding it with traffic, causing significant downtime and operational disruption.
- **Insider threat** : Not all cyber threats come from outside an organization. There is also a risk that trusted individuals with access to sensitive data may inadvertently or maliciously harm the organization.
- **Identity-based attacks** : Most breaches involve compromised identities. Cyber attackers steal or guess user credentials and use them to gain access to an organization's systems and data, often leading to significant breaches.
- **Internet of Things (IoT) attacks** : IoT devices are increasingly targeted, especially legacy devices that lack modern security controls. These attacks can exploit vulnerabilities in connected devices to gain access to broader network systems.
- **Supply chain attacks** : Cybercriminals may target an organization by compromising software or hardware supplied by a third-party vendor. These attacks exploit the trust relationships between vendors and their clients, making them particularly insidious and difficult to detect.
- **Code injection** : By exploiting vulnerabilities in how source code handles external data, cybercriminals inject malicious code into an application. This method allows attackers to execute unauthorized commands or gain control of the application.

Next, we'll focus on how to prepare before incidents occur. Solid preparation – such as setting up detection systems, training staff, and configuring security tools – ensures faster and more effective responses to advanced threats.

Pre-incident preparation

Preparation is crucial to effectively responding to advanced threats. By setting up **robust detection mechanisms**, conducting **regular training**, and ensuring the **proper configuration of security tools**, organizations can significantly enhance their readiness. Ensure all systems and devices are synchronized with a reliable time source through **Network Time Protocol (NTP)**. Accurate

timekeeping is essential for correlating logs and events across different systems during an investigation. Configure NTP on all network devices, servers, and critical systems to maintain synchronized time.

Enable logging on all critical systems, applications, and network devices. Logs provide vital information for detecting and analyzing security incidents. Use a centralized logging solution, such as **Azure Monitor** and **Log Analytics**, to aggregate logs from multiple sources, making it easier to search and analyze data. Implement a formal change management process to ensure that all changes to the IT environment are documented, tested, and approved. This reduces the risk of introducing vulnerabilities through uncontrolled changes. Maintain detailed records of all changes, including who made the change, what was changed, and when it was changed. This information is crucial for investigating incidents.

Apply the **principle of least privilege**, ensuring users have only the access necessary to perform their jobs. Regularly review and adjust permissions as needed. Implement **multi-factor authentication (MFA)** for all user accounts to enhance security and reduce the risk of account compromise. Use separate accounts for different services and systems. Avoid using shared accounts and ensure each service/system has its own unique account. Regularly monitor and audit service and system accounts for unusual activity, such as unexpected logins or changes.

Maintain an **up-to-date inventory** of all IT assets, including hardware, software, and network devices. Knowing what assets you have is essential for effective security management. Classify assets based on their importance to the organization. This helps prioritize security measures and incident response efforts. **Establish alternative communication channels** for use during an incident, especially if primary communication methods are compromised. This could include secure messaging apps, satellite phones, or physical meet-up locations. Develop and regularly update a communication plan that outlines how to use these alternative channels during an incident.

By proactively managing these key areas, organizations can significantly strengthen their security posture. Regularly **review and incorporate new features** and best practices from cloud service providers. This ensures that your security measures are aligned with the latest technological advancements and threat landscapes. Continuously assess and manage the exposure risks of your IT environment. Implement security controls to minimize the attack surface and regularly review access controls and configurations. Conduct **regular training sessions** and incident response simulations to ensure that your team is prepared for real-world threats. These exercises help identify weaknesses and improve the effectiveness of your response strategies.

Incident detection and validation

Once an alert is triggered, the next step involves validating the alert to confirm whether it represents a real threat. This phase includes detailed analysis and cross-referencing with external intelligence

sources. It is very important to know what's normal activity, having some sort of baseline could help analysts tremendously to understand whether the incident is yet another FP or a TP.

Incident detection and validation are fundamental steps in the cybersecurity response lifecycle, ensuring that threats are accurately identified and properly assessed. This process involves leveraging advanced tools and techniques to confirm the presence of a threat, understand its scope, and determine the appropriate response.

Advanced threat detection is the first step in identifying potential security incidents. This involves monitoring various data sources and using sophisticated detection mechanisms to surface risks and potential breaches. **Security tools** that monitor endpoints, identities, networks, applications, and cloud environments help in detecting suspicious activities. **Security professionals** also use threat-hunting techniques to uncover sophisticated threats that might evade traditional detection methods. This proactive approach involves searching for IOCs and **tactics, techniques, and procedures (TTPs)** that suggest a potential threat.

Once an alert is generated, the next step is to **validate its legitimacy**. This involves a thorough analysis of the alert details to understand the context and specifics of the detected threat. Key elements to examine include the type of threat, the affected entities (such as users, devices, or applications), and the activities associated with the alert. Using the **Microsoft Defender XDR portal**, security analysts can investigate the alert to determine whether it represents a genuine threat or an FP.

To enhance the accuracy of threat validation, it is essential to **correlate** the alert data with information from other Microsoft Defender products, such as **Defender for Endpoint** and **Defender for Office 365**. This multi-source correlation helps build a comprehensive picture of the incident, identifying all touchpoints and compromised assets. By integrating data from various sources, security teams can confirm the validity of the threat and understand its full impact on the organization.

Threat intelligence is a critical component of the validation process. Integrating threat intelligence feeds from sources such as the **Microsoft Threat Intelligence Center (MSTIC)** enriches the alert data with information about known threat actors, their TTPs, and IOCs. This context helps in assessing the credibility of the threat and understanding its potential implications. Reputation analysis of IP addresses, domains, and file hashes involved in the alert provides additional insights into whether these entities are associated with known malicious activities.

Advanced technologies such as **machine learning** and **artificial intelligence** play a significant role in the validation process. Behavioral analysis tools leverage machine learning algorithms to analyze user and entity behaviors, identifying anomalies that may indicate a threat. These tools help detect subtle patterns and deviations from normal behavior that might be missed by traditional analysis methods. Additionally, AI models can automatically correlate data across different sources, enhancing the speed and accuracy of threat validation.

To conclude the validation process you may want to ask yourself if the type of event that triggered the alert and incident is a human error, normal circumstances by the system, the time of the day the event occurred, have it happened on another day at the same time, what type of criticality does the system has for the business, does the system have any known and new vulnerabilities – answering those type of questions will help you in the process.

If you find that it is indeed a real incident that doesn't fit the normal activity or baseline, it is important to know what type of next step you can take to eliminate the threat and trigger the incident response process.

Practical steps in the validation process

1. **Alert analysis** : Start by analyzing the alert details in the [Microsoft Defender XDR portal](#) . Look into the type of threat, the entities involved, and the context of the alert.
2. **Correlation** : Use **Advanced Hunting** and **Microsoft Sentinel** to correlate the alert data with logs and events from other Defender products, and other data sources (based on your organization's requirements), to confirm the threat.
3. **Threat intelligence** : Integrate threat intelligence feeds to add context and validate the alert against known threat actors and TTPs.
4. **Behavioral analysis** : Utilize anomaly detection models such as isolation forests or autoencoders to examine user and entity behavior for unusual activities that support the validity of the threat.
5. **Automated correlation** : Use models such as random forests and support vector machines to automatically correlate data across different sources, ensuring a comprehensive validation process.

With validation completed, the next step is to initiate incident response for any confirmed threats identified through MDI alerts. Each alert type points to specific risks within AD, and a targeted response is essential to contain potential security impacts. The following scenarios outline common MDI alerts, along with practical response steps and KQL queries to assist in the investigation and containment process.

MDI incident response scenarios

When MDI alerts are triggered, quickly investigating, validating, and responding is crucial for limiting exposure and impact. Here are some key MDI alert scenarios, suggested response strategies, and KQL queries that can help streamline the investigation.

Scenario 1 – Suspected Suspicious Kerberos Ticket Request

The *Suspected Suspicious Kerberos Ticket Request* alert indicates potential anomalies in the Kerberos authentication process within AD. This alert often highlights suspicious ticket requests or abnormal patterns in the use of Kerberos tickets, which attackers might exploit to gain unauthorized access, escalate privileges, or bypass traditional security controls. Kerberos-based attacks such as pass-the-ticket or golden ticket attacks exploit misconfigurations or vulnerabilities within Kerberos to

impersonate legitimate accounts, posing a significant threat to the integrity of Active Directory environments.

Kerberos tickets grant users access to resources without repeatedly transmitting passwords, which is essential for secure, continuous authentication. However, this system becomes vulnerable if attackers can obtain or forge these tickets. Methods such as pass-the-ticket or golden ticket attacks allow attackers to forge **Ticket Granting Tickets (TGTs)** or misuse **Ticket Granting Service (TGS)** requests to gain prolonged, unauthorized access. These actions often manifest in abnormal ticket request patterns, inconsistent encryption protocols, and unusual logon behaviors, all of which MDI is designed to detect.

Correlation with important event IDs

To investigate this alert effectively, we can correlate this alert with related event IDs and logs across MDI, Defender XDR, and Microsoft Sentinel:

- **Event ID 4624 (Successful Logon)** : High-frequency successful logons, especially across multiple devices or IPs, can indicate forged ticket use or account misuse.
- **Event ID 4768 (TGT Request)** : Elevated TGT request frequency for a single account could signal golden ticket use, where attackers attempt to persistently impersonate high-privilege accounts.
- **Event ID 4769 (TGS Request)** : Suspicious or repeated TGS requests for privileged service accounts or **Service Principal Names (SPNs)** may point to reconnaissance or privilege escalation attempts.
- **Event ID 4771 (Failed TGT Request)** : Frequent failed TGT requests can reveal brute-force or password spray attacks.

Each of these event IDs provides valuable context, helping analysts detect and confirm abnormal Kerberos activity by cross-referencing with MDI's `IdentityLogonEvents` and Sentinel's event logs.

Response and mitigation steps

When responding to a *Suspected Suspicious Kerberos Ticket Request* alert, immediate and methodical investigation is key to understanding the scope and impact of the suspicious activity. The following steps help analysts thoroughly assess the situation and act quickly to prevent further compromise:

- **Audit logon patterns** : Review successful and failed logon events in Defender XDR, looking for concurrent logons or logons from unusual locations. Cross-reference with `IdentityLogonEvents` to identify anomalous activity associated with the account.
- **Analyze ticket details** : Use Sentinel to check TGT and TGS ticket lifetimes. Abnormally long ticket durations or frequent renewals may indicate golden ticket use.
- **Verify encryption standards** : Ensure Kerberos tickets use AES encryption rather than RC4. Weak encryption could expose tickets to brute-force attacks, a common approach in pass-the-ticket scenarios.

Once the initial investigation reveals indicators of suspicious Kerberos activity, it's essential to contain the threat and prevent any potential lateral movement:

- **Isolate affected devices** : Use Defender XDR's isolation features to contain devices associated with suspicious tickets, limiting potential lateral movement.

- **Reset affected account credentials** : Reset passwords and force sign-outs for any impacted accounts, invalidating any active Kerberos tickets.
- **Revise ticket policies** : Review and enforce strict Kerberos ticket policies, including setting appropriate TGT and TGS lifetimes and ensuring sensitive accounts use AES encryption exclusively.

Now, let's examine another scenario.

Scenario 2 – Suspicious Additions to Sensitive Groups

The *Suspicious Additions to Sensitive Groups* alert indicates that an account has been added to a high-privilege AD group, such as Domain Admins, Enterprise Admins, or Administrators. These sensitive groups provide broad permissions across the domain, making unauthorized additions a significant security threat. Attackers commonly target these groups to gain elevated access, potentially allowing them to escalate privileges or carry out other malicious activities across the network.

Sensitive groups in AD confer powerful permissions, so unauthorized modifications to these groups are often signs of privilege escalation attempts. Attackers may add compromised accounts or newly created, unauthorized accounts to these groups to establish persistence and gain greater control. This alert signals unusual changes to these groups, indicating that an adversary might be attempting to expand their access or execute configuration changes that affect the domain.

Correlation with important event IDs

To enhance the investigation, we should correlate this alert with specific event IDs and telemetry across Sentinel and MDI, particularly focusing on indicators of privilege escalation and unauthorized access:

- **Event ID 4728 (Member Added to Global Group)** : This event tracks the addition of users to global groups such as Domain Admins. It is highly relevant for detecting escalated privileges.
- **Event ID 4732 (Member Added to Local Group)** : Logs additions to local groups, which can be critical if sensitive groups on domain controllers or specific servers are modified.
- **Event ID 4756 (Member Added to Universal Group)** : Universal groups, such as Enterprise Admins, have extensive reach, making unauthorized additions to these groups a serious risk.

Response and mitigation steps

Swift response to unauthorized sensitive group changes is crucial. Here's how to proceed:

- **Validate recent group modifications** : Use Sentinel to locate specific group modification events (4728, 4732, 4756) related to sensitive groups. Investigate any accounts added, particularly if they have a history of recent suspicious activity or irregular access patterns.
- **Check authorization of modifying account** : Review the account that made the addition to verify whether the change was authorized. Correlate recent logon events for this account to detect any anomalies in logon times, IPs, or devices.
- **Examine other sensitive group changes** : Use MDI's **IdentityDirectoryEvents** table to detect any other recent sensitive group modifications, which may reveal additional unauthorized changes designed to establish persistence.

READ MORE ABOUT SENSITIVE ENTITIES

For more information on sensitive entities, MDI maintains a list of groups classified as sensitive due to their elevated privileges and potential impact if compromised. You can find the official list and details on these entities in the Microsoft Learn documentation: <https://learn.microsoft.com/en-us/defender-for-identity/entity-tags#default-sensitive-entities>.

To efficiently monitor or even hunt for changes to high-value Active Directory groups, the following KQL query can be an essential tool in your security toolkit:

```
let SensitiveGroupName = pack_array(
    'Account Operators',
    'Administrators',
    'Backup Operators',
    'Domain Admins',
    'Domain Controllers',
    'Enterprise Admins',
    'Enterprise Read-only Domain Controllers',
    'Group Policy Creator Owners',
    'Incoming Forest Trust Builders',
    'Microsoft Exchange Servers',
    'Network Configuration Operators',
    'Power Users',
    'Print Operators',
    'Read-only Domain Controllers',
    'Replicators',
    'Schema Admins',
    'Server Operators'
);
IdentityDirectoryEvents
| where Application == "Active Directory"
| where ActionType == "Group Membership changed"
| extend ToGroup = tostring(parse_json(AdditionalFields).["TO.GROUP"])
| extend FromGroup = tostring(parse_json(AdditionalFields).["FROM.GROUP"])
| extend Action = iff(isempty(ToGroup), "Remove", "Add")
| extend GroupModified = iff(isempty(ToGroup), FromGroup, ToGroup)
| extend Target_Group = tostring(parse_json(AdditionalFields).["TARGET_OBJECT.GROUP"])
| where GroupModified in~ (SensitiveGroupName)
```

This KQL query checks for any membership changes across a specified list of sensitive groups, helping to quickly identify unauthorized additions or removals.

If we confirm unauthorized additions to sensitive groups, it's critical to begin containment and remediation steps immediately to prevent further exploitation of elevated privileges:

- **Remove unauthorized accounts** : Immediately remove any unauthorized accounts from sensitive groups in AD to prevent attackers from leveraging elevated permissions
- **Reset credentials for compromised accounts** : If any compromised accounts were added to sensitive groups, reset their credentials and enforce MFA
- **Isolate affected devices** : Using Defender XDR, isolate devices associated with suspicious logon activity, preventing further potential lateral movement and unauthorized access

Let's turn to our final scenario.

Scenario 3 – Group Policy Tampering

The *Group Policy Tampering* alert signals that suspicious modifications have been made to security-related settings within **Group Policy Objects (GPOs)**. While disabling Windows Defender Antivirus is a common tactic, attackers can tamper with a range of other critical security settings on Windows servers to create additional vulnerabilities or reduce detection capabilities. By altering these configurations, attackers with elevated privileges can facilitate further actions, such as deploying ransomware or moving laterally within the network. This alert is a crucial early warning of potentially significant security breaches.

GPOs allow administrators to manage security settings centrally, controlling configurations for all systems across a domain. Attackers who gain privileged access may target specific security settings within GPOs to weaken defenses, disable protective measures, or establish persistence. The following are common security settings on Windows servers that attackers may tamper with to harm the security posture:

- **Antivirus settings** : Disabling Windows Defender Antivirus or other third-party antivirus solutions to prevent detection of malicious software
- **Firewall settings** : Disabling or reducing firewall protection levels to facilitate inbound and outbound connections without restriction, enabling lateral movement
- **Remote desktop configuration** : Enabling or altering **Remote Desktop Protocol (RDP)** settings to allow remote access, which attackers often use for lateral movement or persistence
- **Audit and logging policies** : Disabling or reducing logging and auditing to minimize traces of their activity, making detection and investigation more difficult
- **PowerShell execution policies** : Modifying PowerShell execution settings to allow unrestricted scripts, facilitating automated attacks or remote code execution
- **User Account Control (UAC)**: Reducing UAC levels or disabling UAC altogether, allowing unauthorized modifications, and reducing prompts for elevation

Suggested steps for investigation

When responding to this alert, we should assess the legitimacy of the change, determine the extent of affected systems, and reverse unauthorized modifications to restore security:

- **Verify the legitimacy of the change** : Start by confirming whether the GPO change was authorized. Review recent changes in AD logs and validate with system administrators to ensure the modification aligns with approved updates.
- **Revert unauthorized security changes** : If the change was unauthorized, immediately restore the GPO to its prior secure configuration. Consider restoring critical security settings, including antivirus, firewall, and audit settings.
- **Assess the impact scope** : Determine which systems are linked to the modified GPO and the scope of changes across the domain, as this will help prioritize containment and restoration activities.

Response and mitigation steps

If tampering is confirmed, follow these steps to restore security:

- **Review recent GPO modifications** : Use Sentinel and MDI to audit recent GPO changes, particularly focusing on those affecting security configurations. Check the account responsible for changes, the specific settings altered, and timestamps.
- **Identify linked systems and impacted security settings** : Determine which systems and user groups the modified GPO applies to and review all security settings that were changed (e.g., firewall rules, antivirus status, audit policies).
- **Look for additional privilege escalation indicators** : Evaluate the behavior of the account associated with the GPO change, checking for unusual privilege assignment or logon activity that might indicate compromised credentials.

For confirmed GPO tampering, act immediately to restore security:

- **Restore security settings to the defaults** : Re-enable Windows Defender Antivirus and any other security features affected, such as firewalls and audit policies, to secure the impacted systems.
- **Revoke suspicious privileges** : If an unauthorized account was involved in the change, remove its elevated privileges, reset credentials, and enforce MFA.
- **Isolate affected systems** : If tampered settings have exposed systems, consider isolating those systems to prevent further exploitation or spread of potential malware.

Each of these MDI scenarios represents a critical aspect of AD security. By promptly addressing suspicious Kerberos activity, sensitive group modifications, and GPO tampering, we can significantly mitigate risks associated with these high-impact alerts. In the following section, we'll dive deeper into the strategic execution of these response measures.

Response strategy and execution

In the rapidly evolving landscape of cybersecurity threats, having a **well-defined and actionable response strategy** is critical for minimizing the impact of security incidents. An effective response strategy involves a series of coordinated steps to detect, analyze, contain, eradicate, and recover from security incidents. The goal is to protect organizational assets, reduce downtime, and maintain stakeholder trust.

Key components of an effective response strategy include the following:

- **Preparation :**
 - Establish clear policies and procedures
 - Train staff in their roles and responsibilities
 - Ensure tools and resources are in place and up to date
- **Detection and analysis :**
 - Utilize monitoring tools to detect anomalies
 - Perform thorough analysis to understand the nature and scope of the incident
 - Gather evidence and document findings
- **Containment :**

- Implement immediate measures to prevent the incident from spreading
- Short-term containment to stop the breach
- Long-term containment to ensure the threat is fully neutralized

- **Eradication :**

- Identify and remove the root cause of the incident
- Clean affected systems and ensure they are free from malicious code
- Validate the removal through testing

- **Recovery :**

- Restore systems and services to normal operation
- Monitor systems for any signs of lingering issues
- Conduct a thorough review to ensure all remediation steps were effective

- **Communication :**

- Maintain clear communication with stakeholders throughout the incident
- Provide timely updates to affected parties
- Ensure transparency to maintain trust and comply with regulatory requirements

Next, we will delve into the SOAR tools available in **Microsoft Cloud**. While **Azure Logic Apps** is commonly used – primarily because it's integrated into the current architecture of Defender XDR and Microsoft Sentinel – we are not restricted to relying solely on it. We can integrate other, more scalable tools to enhance our security orchestration and automation capabilities.

SOAR tools in Microsoft Cloud

In the context of the Microsoft Cloud, several tools can be used to build an effective **Security Orchestration, Automation, and Response (SOAR)** solution. Each tool has its unique strengths and use cases, enabling comprehensive and automated security operations.

Power Automate

Power Automate is a versatile tool that allows you to automate repetitive tasks and workflows across multiple applications and services. It integrates seamlessly with various Microsoft products and external services, enabling the creation of complex automation workflows without writing any code. You can use Power Automate to do the following:

- Automate data collection from different sources
- Trigger alerts and notifications based on specific conditions
- Integrate with Microsoft Teams for collaborative incident response

Logic Apps

Logic Apps offers robust automation capabilities, allowing you to create complex workflows that integrate with a wide range of services and applications. In the context of Defender and Sentinel, Logic Apps can be used to do the following:

- Automate incident response workflows
- Integrate with third-party security tools and services
- Perform automated threat hunting and data enrichment

Azure Automation

Azure Automation provides process automation and configuration management capabilities. It helps automate frequent, time-consuming, and error-prone cloud management tasks. You can use Azure Automation to do the following:

- Automate the deployment of resources in Azure
- Create runbooks for incident response
- Integrate with other Azure services to streamline operations

Azure Functions

Azure Functions allows you to run small pieces of code (functions) in the cloud without managing infrastructure. It is ideal for automating responses to specific events. Use Azure Functions to do the following:

- Execute code in response to triggers from other Azure services
- Perform custom processing and analysis of security data
- Integrate with APIs and other external systems for extended functionality

Building automation

Collaboration between **Security Operations (SecOps)** and IT professionals, including **Development Operations (DevOps)** teams, is important for building robust security frameworks. This integration leverages the strengths of both teams to create a seamless and effective security operation. Here's how this collaboration can be implemented, focusing on automation, detections, and DevOps methodologies.

DevOps teams are skilled in creating and managing **continuous integration and continuous deployment (CI/CD)** pipelines. By integrating these checks into these pipelines, DevOps can help SecOps teams automate the creation of detections and automation playbooks.

To effectively build automation within security operations, leveraging DevOps practices like CI and CD is essential. These methodologies enable the automation of development, testing, and deployment

processes for security detections and responses. The following sub-sections explore how continuous integration and continuous deployment contribute to building a more automated and resilient security framework.

Continuous integration (CI)

In the **continuous integration (CI)** process, DevOps teams play a crucial role by implementing automated testing for code quality and security through linting and other checks. These CI tests ensure that any changes to the codebase do not introduce errors and adhere to best practices:

- **Linting** : Automated linting tools check for syntax errors and enforce coding standard, such as YAML files, KQL queries, or other files, ensuring they are efficient and free from common mistakes. This helps maintain a high quality of detections and reduces FPs or inactivation of the analytic rule.
- **Unit testing** : This involves creating test cases for individual components of the KQL queries to ensure they work as expected. Unit tests help detect issues early in the development process, ensuring that each part of the query functions correctly.
- **Security checks** : Perhaps not related to this type of project, but relevant in many other types of DevOps projects. CI pipelines can include security-specific tests that analyze the code for potential vulnerabilities or insecure coding practices. This proactive approach helps in identifying and mitigating risks before they can be exploited.

With these CI practices, organizations ensure that security detections and code are consistently tested and maintained at a high standard before deployment. This integration enhances the reliability of security measures and supports the overall goal of building automation within SecOps.

Continuous Deployment (CD)

The **continuous deployment (CD)** aspect of DevOps ensures that validated and tested configurations, such as KQL queries and automated playbooks, are deployed seamlessly into the production environment. This continuous improvement cycle ensures that the security infrastructure is always up to date with the latest detections and response strategies. Key components of continuous deployment include the following:

- **Automated deployments** : CD pipelines automate the deployment of KQL detections and SOAR playbooks into production. This reduces the time it takes to implement new security measures and ensures consistency across environments.
- **Version control and rollback** : By using version control systems, such as Git, DevOps teams can manage changes to security configurations, allowing for easy rollback in case of issues. This ensures stability and reliability in the security operations environment.

Implementing CD practices allows SecOps teams to rapidly and reliably update security infrastructure. This seamless deployment process is a critical component of building automation, as it minimizes manual interventions and accelerates the response to emerging threats.

Utilizing KQL detections

Building on the automation provided by CI/CD pipelines, the effective use of **Kusto Query Language (KQL)** for detections is another key element in enhancing security operations. KQL enables the

creation of precise and efficient queries that are essential for threat detection within Defender XDR and Microsoft Sentinel. Following are the main uses of KQL detections:

- **Creating and refining KQL queries** : KQL is a powerful tool for analyzing large datasets and creating custom detections in **Microsoft Sentinel** and **Defender**. Security analysts can write KQL queries to detect specific threats and anomalies in the data. However, these queries need to be precise and efficient to avoid performance issues and false positives.
- **Collaboration on KQL implementations** : DevOps teams can assist security analysts by verifying the syntax and performance of KQL queries. They can integrate these queries into the monitoring systems and ensure that they run efficiently within the existing infrastructure. This collaboration ensures that detections are accurate and do not negatively impact system performance.
- **Automating detection and response** : Once the KQL detections are validated, DevOps teams can help automate the response workflows using SOAR tools. For instance, when a specific threat is detected, the system can automatically trigger a series of actions, such as isolating affected systems, notifying relevant stakeholders, and logging the incident for further analysis.

By effectively utilizing KQL detections and integrating them into automated workflows, organizations enhance their ability to proactively identify and mitigate potential threats. However, not all incidents can be anticipated or automatically resolved. High-stakes situations require a structured and immediate response to minimize impact. This necessitates a robust incident response action plan that complements automated detection mechanisms.

Incident response – an action plan for high-stakes situations

When dealing with high-stakes incidents, a robust action plan is essential to manage the situation effectively and minimize damage. This section outlines the critical steps in developing and executing an incident response action plan.

Building an incident response team

The foundation of an effective incident response strategy is a well-structured **Incident Response Team (IRT)**. This team should include individuals with a range of skills and responsibilities, each contributing to the detection, analysis, and mitigation of security incidents.

The IRT should include members from various departments, such as IT, legal, human resources, and public relations, each bringing unique expertise to address different aspects of an incident. Clearly defining the roles and responsibilities of each team member ensures that everyone knows their duties and can act swiftly and effectively during an incident. Key roles in the IRT typically include an incident response manager, analysts, forensic experts, system/network/cloud administrators, and communication officers. One of the most important skills for these roles is the ability to work as a team and communicate effectively with each other.

The IRT must also have the authority to make critical decisions quickly, with clear protocols established for escalation and decision-making during incidents.

IRT models are essential for structuring how organizations prepare for, identify, and respond to cybersecurity incidents. Effective team models ensure that an organization can handle incidents promptly and efficiently, minimizing damage and facilitating recovery. The **National Institute of Standards and Technology** (NIST) outlines several models for incident response teams in its Special Publication 800-61 Revision 2, each with unique benefits and considerations.

Understanding the various IRT models allows organizations to select the structure that best fits their specific needs and operational contexts. In the following sections, we will examine these models in detail to provide insights into how they can enhance your organization's incident response capabilities.

Centralized IRT

In the centralized model, a single team is responsible for incident response across the entire organization. This model is typically used in smaller organizations or those with a centralized IT infrastructure:

- **Structure and composition :**
 - **Core team** : The team consists of a core group of specialists, including incident handlers, forensic analysts, and communication officers
 - **Leadership** : Led by an incident response manager who coordinates activities and makes strategic decisions
- **Advantages :**
 - **Unified response** : Ensures a consistent approach to incident handling
 - **Resource efficiency** : Reduces the need for multiple teams and resources
 - **Clear communication** : Simplifies communication channels within the organization
- **Challenges :**
 - **Scalability** : May struggle to manage incidents in large, distributed environments
 - **Overload risk** : The team may become overwhelmed during multiple or large-scale incidents

Distributed IRT

The distributed model involves multiple IRTs, each responsible for specific geographic locations or business units. This model is suitable for larger organizations with diverse operations:

- **Structure and composition :**
 - **Multiple teams** : Each team operates independently within its domain but follows a common framework and protocols

- **Regional or business unit focus** : Teams are aligned with specific regions, business units, or functional areas
- **Advantages :**
 - **Scalability** : Capable of handling incidents across large, complex organizations
 - **Specialization** : Teams can develop expertise in specific areas relevant to their domain
 - **Local presence** : Facilitates quicker response times and better understanding of local contexts
- **Challenges :**
 - **Coordination complexity** : Requires robust coordination mechanisms to ensure consistency and communication across teams
 - **Resource duplication** : Potential for duplicated resources and efforts

Coordinated IRT

In the coordinated model, there are multiple incident response teams, but they are centrally coordinated. This model combines elements of both centralized and distributed models and is suitable for large organizations with complex infrastructures:

- **Structure and composition :**
 - **Local teams** : Operate independently within their domains but adhere to guidelines and policies set by a central coordination body
 - **Central coordination** : A central team oversees and coordinates the efforts of local teams, ensuring consistency and providing additional support as needed
- **Advantages :**
 - **Balanced approach** : Combines local expertise and responsiveness with centralized oversight and coordination
 - **Resource optimization** : Efficiently utilizes resources by leveraging both local and central capabilities
 - **Consistency** : Maintains uniformity in incident response procedures across the organization
- **Challenges :**
 - **Coordination overhead** : Requires effective communication and coordination mechanisms
 - **Complex governance** : Needs a robust governance framework to manage the interaction between local and central teams

Outsourced IRT

Some organizations opt to outsource their incident response capabilities to external providers. This model can be beneficial for organizations lacking internal expertise or resources:

- **Structure and composition :**

- **External service providers** : Third-party vendors provide incident response services, either fully or in a hybrid model with internal teams
- **Service agreements** : Defined by detailed SLAs and contracts
- **Advantages** :
 - **Expertise and experience** : Access to specialized skills and advanced resources
 - **Cost efficiency** : Can be more cost-effective than maintaining an in-house team, especially for smaller organizations
 - **Scalability** : Providers can offer scalable solutions that adjust to the organization's needs
- **Challenges** :
 - **Dependency** : Reliance on external providers can pose risks if not managed properly
 - **Response time** : Potential delays in response time due to the provider's other commitments
 - **Control and oversight** : Less direct control over the incident response process

Hybrid IRT

A hybrid model integrates internal IRTs with external support. This model allows organizations to maintain core incident response capabilities while leveraging external expertise for specialized tasks or large-scale incidents:

- **Structure and composition** :
 - **Internal core team** : Manages day-to-day incidents and maintains in-house knowledge
 - **External support** : Engages external providers for advanced analysis, large-scale incidents, or specific expertise
- **Advantages** :
 - **Flexibility** : Combines the strengths of both internal and external resources
 - **Scalability and expertise** : Scales to handle larger incidents and accesses specialized skills when needed
 - **Cost management** : Optimizes costs by balancing in-house and outsourced capabilities
- **Challenges** :
 - **Integration complexity** : Requires effective integration and coordination between internal and external teams
 - **Clear agreements** : Necessitates clear SLAs and agreements to manage expectations and responsibilities

After examining the various IRT models – centralized, distributed, coordinated, outsourced, and hybrid – it is clear that the structure of your IRT plays a significant role in how effectively your organization can respond to security incidents. However, the team's success is not solely dependent on its composition; it also relies heavily on having a well-defined strategy in place. This brings us to the essential component that guides the actions of any IRT: the **Incident Response Plan (IRP)**.

Incident Response Plan (IRP)

As you might understand by now, being prepared for potential threats must be one of any organization's top priorities. This preparation involves having a well-defined and structured plan in place to respond to any incidents that may occur. This plan, known as an IRP, is a set of instructions that guide an organization in identifying, responding to, and recovering from security incidents. The goal is to handle the situation in a way that limits damage and reduces recovery time and costs.

The NIST provides a standard for incident response in its Special Publication 800-61 Revision 2. This standard outlines a process divided into four main phases: **Preparation**, **Detection and Analysis**, **Containment, Eradication, and Recovery**, and **Post-Incident Activity**. Let's delve into each of these phases in more detail:

- **Preparation** : This is the phase where an organization establishes and maintains an incident response capability. It includes creating policies that define the scope of incident response, plans that provide a roadmap for implementing these policies, and procedures that offer low-level, tactical, step-by-step guidance on how to respond to incidents. For example, an organization might establish a policy requiring all employees to report potential security incidents, such as phishing attempts or lost devices.
- **Detection and Analysis** : In this phase, potential incidents are identified. This could be through an **intrusion detection system** (**IDS**) flagging suspicious network traffic or a user reporting a slow computer. Analysis is performed to check whether an incident has occurred and to understand its nature. For instance, a series of failed login attempts from a foreign IP address might be detected, leading to the analysis and identification of a potential brute-force attack.
- **Containment, Eradication, and Recovery** : Once an incident is confirmed, steps are taken to limit the damage it can cause (containment), remove its cause (eradication), and restore affected systems to normal operation (recovery). For example, containment might involve disabling compromised accounts or changing their permissions. Eradication could mean identifying and removing the source of the threat, such as a phishing attack or malware. Finally, recovery involves restoring the affected systems to normal operation, re-enabling user accounts, and improving security measures to prevent future incidents.
- **Post-Incident Activity** : After an incident is handled, a thorough review is conducted to understand what happened and why, and to learn lessons for the future. This could involve an RCA to determine how the incident occurred and an evaluation of the incident response to identify any areas for improvement. For instance, if an incident was caused by a phishing email, the post-incident activity might involve user education to prevent similar incidents in the future. As the NIST framework suggests, holding a **lessons learned** meeting with all involved parties is crucial to improving the incident handling process.

Remember, the goal of an IRP is not just to react to incidents, but to handle them in a structured and efficient manner that minimizes damage and recovery time. It's also about learning from these incidents to improve future response efforts and enhance the overall security posture.

As discussed in the previous sections, the IRP should be *regularly reviewed and updated* to reflect changes in the threat landscape and organizational structure. It's crucial not to adopt a set-and-forget approach to these important topics. Consistently updating and refining your incident response strategies ensures preparedness and enables proactive rather than reactive measures. Being well prepared is always better than scrambling to respond in the face of a crisis.

NOTE

In addition to the NIST framework, consider exploring other incident response frameworks such as ISO/IEC 27035, the SANS Incident Handler's Handbook, CERT Guidelines, and COBIT 5 for comprehensive guidelines and best practices. Each offers unique approaches and valuable insights for effective incident management.

Summary

In this chapter, we have focused on the essential processes of managing alerts and incidents within MDI. We explored how the MDI alert system functions, providing insights into the different types of alerts and their classifications. We delved into the initial triage process, highlighting the steps to assess and categorize incidents effectively.

Furthermore, we examined the automation capabilities available within the Microsoft Cloud, including Power Automate, Logic Apps, Azure Automation, and Azure Functions. These tools enable streamlined and efficient incident response workflows.

Finally, we discussed the importance of having a structured IRP and building a capable IRT. Emphasizing the need for regular reviews and updates, we underscored the significance of being prepared and proactive in the ever-evolving cybersecurity landscape.

Next, we will delve into the strategic use of MDI action accounts, focusing on the configuration, security best practices, and the critical role it plays in enhancing automated threat response.

8

Utilizing MDI Action Accounts Effectively

In this chapter, we will focus on the strategic use of **Microsoft Defender for Identity (MDI) action accounts** to boost security and operational efficiency. We'll start by walking through the essential steps of configuring these accounts together, highlighting best practices (I know, some of you may cringe at the term, but hey, we need a solid foundation, right?) to ensure they're secure. You'll learn how to strike the perfect balance between permissions and security protocols, so your action accounts are both functional and well protected against potential threats.

After that, we'll switch gears and get into the real-world stuff – practical examples of when action accounts take the spotlight in automated threat responses. Through case studies, you'll see firsthand how these accounts can be your secret weapon in detecting and mitigating various security challenges, from credential theft to lateral movement attacks. We'll also touch on how action accounts can streamline security operations, helping you respond to incidents with speed and efficiency – because nobody likes scrambling in a crisis.

By the end of the chapter, you'll have a solid grasp of how to configure and secure MDI action accounts. And let's not forget the best part – automated responses that have your back when you need them most.

In this chapter, we're going to cover the following main topics:

- Configuring and securing action accounts
- Real-world scenarios and use cases

Technical requirements

Completion of [Chapter 2](#) and [Chapter 3](#) is a prerequisite for this chapter. As a reminder, you also require the following:

- A Microsoft tenant
- A Microsoft subscription that includes MDI, such as Microsoft 365 E5 or Microsoft 365 E3 + E5 Security
- Basic Microsoft 365 knowledge
- Active Directory knowledge
- A virtual or physical server environment with Active Directory installed and configured:
 - **Optional :** Active Directory Federation Services and Active Directory Certificate Services installed and configured
- Basic PowerShell knowledge:

- Windows PowerShell 5.1 or PowerShell 7.4 or later
- Make sure your server has one of these versions, or later, installed

All the code examples for this chapter can be found on GitHub at

<https://github.com/PacktPublishing/Microsoft-Defender-for-Identity-in-Depth/tree/main/Chapter08>

Configuring and securing action accounts

MDI action accounts are the backbone of automated responses within MDI. However, their effectiveness is deeply tied to how well they are configured and secured, or as per the default mode, impersonating the **LocalSystem** account of a **domain controller**. In this section, we will cover the foundational aspects of setting up these accounts, ensuring they are optimized for both security and functionality.

Before diving into the technicalities, let's take a moment to understand what action accounts actually are and why they're so crucial for us.

Understanding action accounts – what are they and why do they matter?

In simple terms, action accounts are specialized accounts used by MDI to perform responses in our **Active Directory** environment. These accounts operate with elevated privileges, making them powerful tools but also potential targets if not properly secured.

The supported actions that can be performed by action accounts in your Active Directory environment are currently the following:

- **Force password reset**, which prompts the user to change their password at the next logon (only available for Active Directory and hybrid users)
- **Disable/enable** user accounts

As you can see in the following screenshot from the **Defender XDR portal**, we can take some actions on a domain account. You can also see that in this case, this account is not synchronized to Entra ID (see *Figure 8.1*).

The screenshot shows the Microsoft Defender XDR interface for managing a user account. At the top, the account name 'localAdmin' is displayed, along with a status indicator showing 'Enabled'. Below the account name, there are two buttons: 'SENSITIVE' (red) and 'NEW' (gray). A horizontal ellipsis menu is open, listing several actions: 'Force password reset', 'View related activity', 'View owned files', 'View files shared with this user', and 'View related incidents'. On the left side of the interface, there are sections for 'Overview' (selected), 'Policies', 'User threat', 'Open incidents' (1), 'Investigation priority' (Not available), 'Active alerts' (1), and 'Entra ID risk level' (Not available).

Figure 8.1 – Actions on accounts in Defender XDR

But why do action accounts matter? Because in the world of cybersecurity, speed and accuracy are everything. Action accounts enable automated responses that can neutralize threats before they escalate, minimizing damage and keeping your organization's infrastructure secure. Without well-configured action accounts, your MDI deployment might as well be a fancy alarm system that rings but doesn't call for help.

Previously, the main challenge centered around managing on-premises identities from the cloud and specifically from the older Defender portal. In contrast, handling cloud identities in **Entra ID** (formerly known as Azure Active Directory) is relatively straightforward because of the built-in integration that enables us to easily disable a user, mark the account as compromised, enforce **Multi-Factor Authentication (MFA)**, or reset the password. However, complications arise when an identity exists in both Entra ID and Active Directory. For example, if you disable a user in Entra ID, this change can be overwritten during the next synchronization with the on-premises environment, leading to potential security gaps.

We are still seeing (as of 2024) many attacks against on-premises environments where the goal for many threat actors is to gain initial access and escalate privileges in order to move laterally within the network, ultimately aiming to exfiltrate sensitive data, deploy ransomware, or establish persistent control over critical systems. A common example of this is **credential phishing attacks**. Despite the

availability of highly effective countermeasures such as MFA, **Safe Links** (a feature in Microsoft Defender for Office 365), and other security tools, incidents still occur. We often see alerts from tools such as Microsoft Defender when a user clicks on a malicious link, potentially compromising their credentials.

Another example is if Defender XDR identifies an ongoing attack (it gathers trillions of signals), such as an active ransomware campaign or an advanced sophisticated attack, with high confidence, it will be disrupted by the **Automatic Attack Disruption** feature. Actions such as disabling a user require specific permissions on the action accounts. As you read further, you'll see the importance of correctly configuring these accounts. Using LocalSystem or appropriately managed **Group Managed Service Accounts (gMSAs)** helps ensure that Attack Disruption can perform these actions efficiently, maintaining effective incident response capabilities without unintentional limitations.

As we just learned, in the past, our options for responding to such incidents where we had on-premises user accounts were limited. We could only mark the user as compromised in Entra ID, setting their risk level to high and relying on a configured user risk policy to force a password reset at the next logon. As can be imagined, there were significant gaps in our defense because of a lack of proper response.

Let's discuss the type of account that will be our bridge between the cloud and on-premises.

Choosing between system or custom gMSAs

The option to use the domain controller's system account was introduced on October 30th, 2022. Now, setting up a gMSA is only necessary if your organization's specific structure requires it. The decision between using a **system account** or a **custom gMSA** depends on your organization's structure, tiering, **Role-Based Access Control (RBAC)**, and whether you're working with a **Managed Security Service Provider (MSSP)**. For instance, if you need to delegate permissions to only affect accounts within certain **Organizational Units (OUs)** within Active Directory, a custom gMSA is the way to go.

This setup is especially beneficial if you have an MSSP partner handling your security through a **Managed Detection and Response (MDR)** service. With custom gMSAs, you can control exactly which accounts your MSSP can take action on, maintaining tight security boundaries while still enabling effective threat response. From an MSSP perspective, these features are invaluable because they allow for quick response to threats without requiring direct access to the customer's environment, such as through legacy VPN solutions. But even if your security operations are managed internally, having these options gladly available within the MDI portal is a must for efficient and effective incident response.

Another aspect to consider is that Active Directory often contains sensitive accounts, such as those protected by the **AdminSDHolder** object, and by using the **LocalSystem** account, you risk giving it

extensive, unrestricted access to the entire Active Directory environment. This can pose significant security risks, especially if the system is compromised. By using a gMSA instead, you can limit access to only the necessary permissions and specific OUs, thereby protecting critical accounts while still ensuring that MDI can effectively perform its automated response actions.

NOTE – ADMINSDHOLDER AND LOCALSYSTEM

AdminSDHolder is a special Active Directory object that protects high-privilege accounts, such as those in the Domain Admins group, by automatically resetting their permissions every hour to maintain security.

LocalSystem is a built-in Windows account with extensive privileges on the local machine and network access using the machine's credentials.

So, it all boils down to that you need to decide which account you will use for MDI action accounts. In the meantime, we will start configuring gMSA in the next section.

Best practices for action account configuration – getting it right the first time

When it comes to configuring action accounts, getting it right from the start is essential.

Misconfiguration can lead to either a lack of necessary functionality or, worse, an open door for attackers. The first step, as discussed earlier, is to determine the scope of what your action accounts can accomplish.

A key best practice (that you perhaps have heard of too many times) is to adhere to the principle of least privilege. This means granting each action account only the minimum level of access necessary to perform its tasks, which reduces the attack surface and makes it harder for malicious actors to exploit these accounts.

When deciding how to structure your MDI action accounts, it's also important to consider whether segregating duties by creating multiple accounts for different tasks is necessary. For instance, you might use separate accounts for automated investigations versus containment actions if your environment's complexity requires it. However, the focus should remain on ensuring that each account, whether singular or multiple, is configured with the minimal permissions needed and is closely monitored. This approach limits the impact if an account is compromised, reducing the overall risk to your environment.

The required permissions and properties the gMSA needs to be able to force password reset and disable a user in Active Directory are the following:

- **Permission :**

- Reset password

- **Property :**

- **Read pwdLastSet**
- **Write pwdLastSet**
- **Read userAccountControl**
- **Write userAccountControl**

Here's a quick step-by-step guide to finding the dialog box and setting the required permissions and properties for our gMSA:

1. On your domain controller, open **Active Directory Users and Computers**.
2. Right-click on an optional **OU** and then click on **Properties**.
3. Click on the **Security** tab and then the **Advanced** button.
4. In the **Advanced Security Settings for <OU name>** dialog box, click on **Add**.
5. In the **Permission Entry for <OU name>** dialog box, click on **Select a principal**:
 - (Recommended) If you had prepared a group, search for the group name.
 - (Not recommended) If you just want to add your gMSA, click on **Object Types** and select **Service Accounts**. Otherwise, you will not be able to find the account.
 - After you have found the object (group or service account), click on **OK**.
6. Change **Applies to** to the **Descendant User objects** value (located at the bottom of the drop-down list).
7. Then, find and select the following:
 - **Permissions** : **Reset password**
 - **Properties** : Read and write **pwdLastSet**

In the advanced security settings for the chosen OU, in the **Permission Entry** dialog box, we can set the required permission and properties (see *Figure 8.2*).

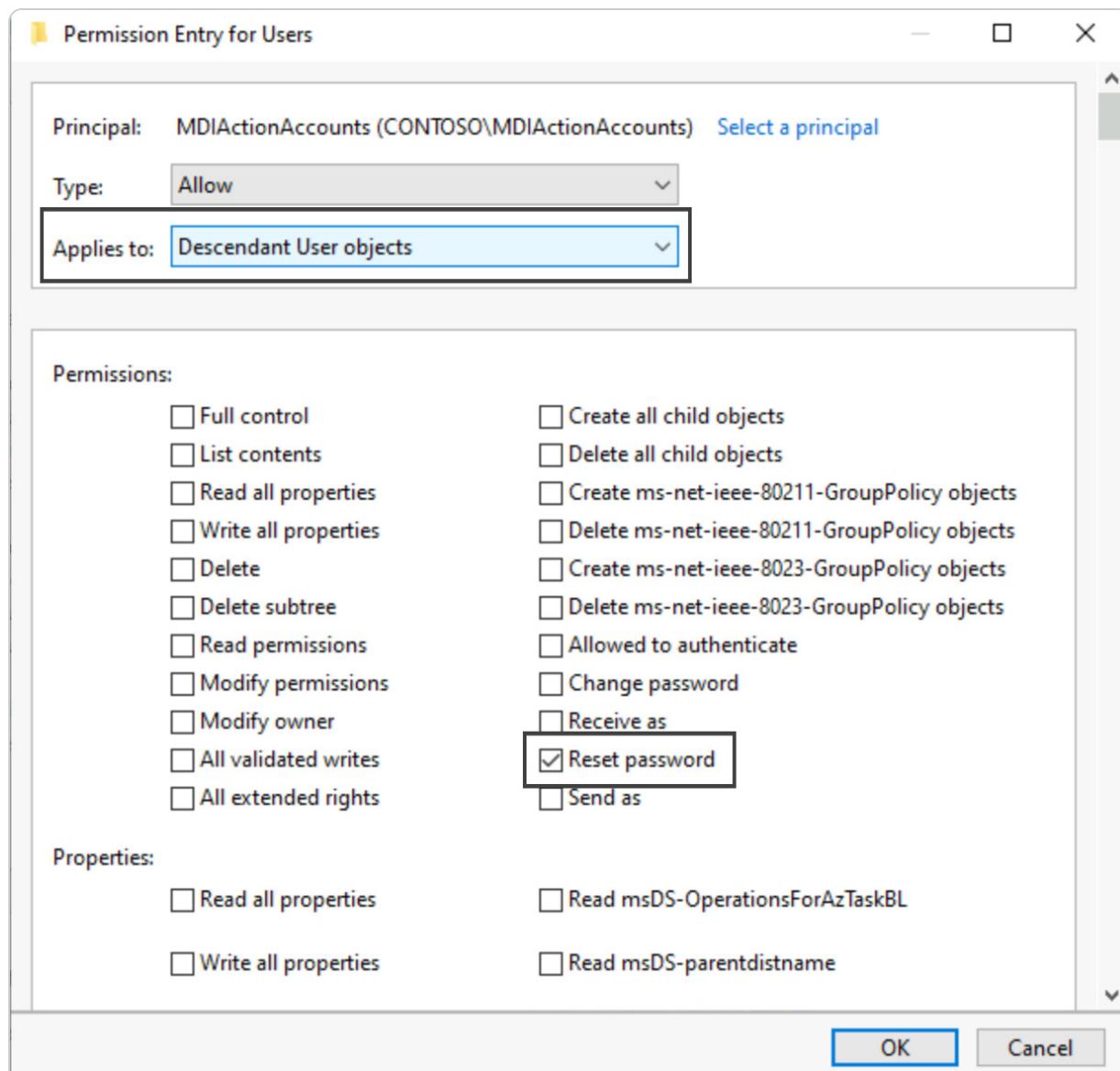


Figure 8.2 – Setting Reset password permission

For the properties, we need to scroll down quite far to find the `pwdLastSet` properties (see *Figure 8.3*).

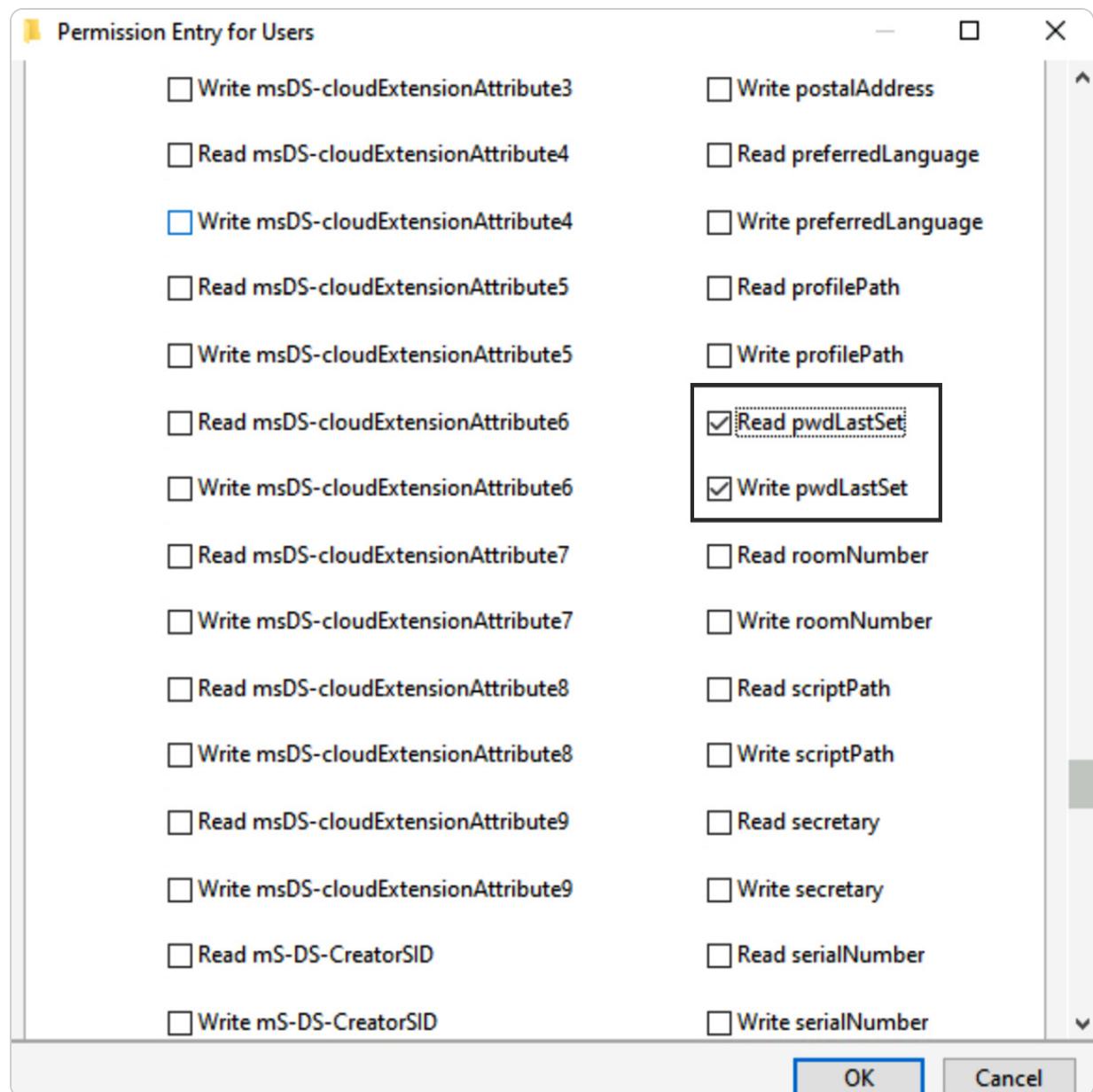


Figure 8.3 – Setting pwdLastSet properties

Setting these permissions and properties in the **Active Directory Users and Computers** snap-in can be challenging, so I'll also show you how to create the gMSA using **PowerShell** as an alternative. The following commands assume you have the Active Directory module installed and the necessary permissions:

1. Log in to your domain controller and start Windows PowerShell from the Start menu.
2. Import the Active Directory module:

```
Import-Module ActiveDirectory
```

3. Define the variables:

```

$accountname = "MDIACTIONGMSA"
getDescription = "Microsoft Defender for Identity Action Account"
$accountdns = $accountname + "." + ((Get-ADDomain -Current
LocalComputer).DNSRoot)
$accountDN = "CN=" + $accountname + ",CN=Managed Service Accounts," + ((Get-
ADDomain -Current LocalComputer).DistinguishedName)
$groupname = "MDIActionAccounts"
$grouppath = "OU=Groups,OU=Contoso" + "," + ((Get-ADDomain -Current
LocalComputer).DistinguishedName)
$SamAccountName = "MDIActionAccounts"
$groupDN = "CN=" + $groupname + "," + $grouppath
$assignOU = "OU=Users,OU=Contoso" + "," + ((Get-ADDomain -Current
LocalComputer).DistinguishedName)
$NETBIOSwGroupName = ((Get-ADDomain -Current LocalComputer).NetBIOSName) + "\\" +
$groupname

```

4. Create a new MDI action account group and gMSA:

```

New-ADGroup -GroupCategory Security -GroupScope DomainLocal -Name $groupname -
Path $grouppath -SamAccountName $SamAccountName
New-ADServiceAccount $accountname -Description $description -DNSHostName
$accountdns -PrincipalsAllowedToRetrieveManagedPassword $groupname -
KerberosEncryptionType AES256

```

5. Add the gMSA to the group to protect the group from accidental deletion:

```

Set-ADObject -Identity $groupDN -ProtectedFromAccidentalDeletion $true
Set-ADGroup -Add:@{'Member'=$accountDN} -Identity $groupDN

```

6. Assign permissions to the group:

```

$params = @("$assignOU", "/I:S", "/G", "$NETBIOSwGroupName`:WP;pwdLastSet;user")
dsacl.exe $params
$params = @("$assignOU", "/I:S", "/G",
"$NETBIOSwGroupName`:WP;userAccountControl;user")
dsacl.exe $params
$params = @("$assignOU", "/I:S", "/G", "$NETBIOSwGroupName`:CA;Reset
Password;user")
dsacl.exe $params
$params = @("$assignOU", "/I:S", "/G", "$NETBIOSwGroupName`:WP;member;group")
dsacl.exe $params

```

7. You should get some output and the **The command completed successfully** message.

In **Defender XDR Unified RBAC**, we cannot say that Tier 1 analysts can only use one action account, because they may have the **Response (Manage)** permission, which uses the gMSA configured for the domain (see *Figure 8.4*). Add more action accounts (gMSAs) when you have multiple forests that need to be protected with MDI. Here's what the **Response (Manage)** permission looks like in Defender XDR Unified RBAC.

Security operations

Select the permissions in this group to users who perform security operations and those who respond to incidents and advisories.

Clear all permissions

All read-only permissions

All read and manage permissions

Select custom permissions

Security data

Read-only

Select all permissions

Select custom permissions

Take response actions, approve or dismiss pending remediation actions, and manage blocked and allowed lists for automation

Response (manage) (i)

Basic live response (manage) (i)

Advanced live response (manage) (i)

File collection (manage) (i)

Figure 8.4 – Unified RBAC – Security operations custom permissions

As you may already know, the **Defender XDR Attack Disruption** feature requires that the advanced auditing policy is configured on your domain controllers for MDI to fully enable this functionality. Since we have already provided a detailed guide on configuring the advanced auditing policy in *Chapters 2 through 4*, we won't repeat those instructions here. However, it's crucial to verify that this policy is properly set up on your domain controllers to leverage the full capabilities of the Attack Disruption feature. The importance of monitoring the MDI configuration is indeed crucial for the ongoing protection of our identities. When we have the events needed for MDI, we also want to make sure that our action accounts are valid and configured correctly with the necessary permissions. Also,

of course, the main component, the MDI sensor itself, needs to be installed on all domain controllers, and the communication toward the MDI service in the cloud needs to be intact.

MDI ACTION ACCOUNT IN MULTI-FOREST ENVIRONMENTS

Configuring gMSAs in a multi-forest setup can add significant complexity to the deployment of MDI sensors. To streamline this process, it's recommended to use the sensor's local system account, which runs in the SYSTEM context. This configuration grants the account full control over objects in Active Directory.

However, if you need to use a gMSA, be aware that in a multi-forest setup, your gMSA must either be trusted across all forests or configured as a separate account in each untrusted forest. A single gMSA can simplify cross-forest management in environments with mutual trust, whereas separate gMSAs help enforce boundary controls in environments without a two-way trust.

Now, let's add the gMSA to the MDI action account settings:

1. Sign in to security.microsoft.com with Security Administrator or an equivalent account that has permission to change the MDI settings.
2. Go to **System > Settings > Identities**.
3. Click on **Manage action accounts**.
4. Then, click on **Manually configure your management accounts**.
5. Click on **+ Add credentials**.
6. In the **Add credentials** window, configure the following:
 - **Account name** : Type the gMSA name. Don't forget the \$ sign, for example, **MDIACTIONACCOUNT\$** .
 - **Domain** : Type in your Active Directory domain name, for example, **contoso.local** .
 - Click on **Save** .

Please refer to the following screenshot for a successful gMSA action account configuration (see *Figure 8.5*).

Microsoft Defender for Identity

This list contains credentials that sensors can use to perform actions on on-premises Active Directory users, such as disable user, reset user password and more. Configure an action account so remediation actions can be taken manually or automatically. [Learn more](#)

Automatically use the sensor's local system account
 Manually configure your management accounts

Export Add credentials 1 item Search

Filter Reset Filters

Domain: Any Group managed service account: Any

Account	Domain	Group managed service account
<input type="checkbox"/> MDIACTIONGMSA\$: contoso.local	True

Figure 8.5 – Manage action accounts page

We have now successfully configured our MDI action account with the gMSA type of object and given that account permissions to help us remediate detected threats in our environment. Now, I want us to discuss how we can protect these gMSAs and how to detect whether someone is poking around on those accounts.

Security measures – protecting your action accounts from compromise

Now that we've set the foundation with proper configuration, let's talk about securing these accounts. Action accounts, due to their elevated privileges, are prime targets for attackers. Therefore, it's imperative to implement robust security measures to safeguard them.

IMPORTANT – SECURE GMSA

gMSA attacks often happen because of misconfigurations, so it's important to be extra careful with permissions. Make sure that only essential users or machine accounts have access to retrieve managed passwords. Remember, those accounts and the hosts using them need solid protection too. If any part of this chain – whether it's a user, a computer, or a managed password – gets compromised, it could put your entire domain and forest at risk.

*Keep a close eye on your Windows event logs for any signs of **golden gMSA attacks**. A golden gMSA attack occurs when an attacker gains control over a gMSA and can use its credentials to move laterally within a network. Since gMSAs can access sensitive resources, compromising them can lead to full domain compromise. If detected, create a **new root Key Distribution Service (KDS) key** and set up **new gMSAs** to prevent attackers from continuing to exploit managed passwords, as they can potentially crack them indefinitely. Acting quickly is critical.*

Read the guide from Microsoft on how to recover from a golden gMSA attack and how to roll the keys when a person with domain administrator resigns from the company: <https://learn.microsoft.com/en-us/troubleshoot/windows-server/windows-security/recover-golden-gmsa>

[from-golden-gmsa-attack](#).

Monitoring and logging are crucial. Keep a close eye on the activity of action accounts (gMSA), looking for any signs of unusual behavior that could indicate an attempted breach.

Here's a short description of how to defend, monitor, and detect a golden gMSA attack:

- **Defend** : To protect against golden gMSA attacks, enforce **strict access controls** on who can retrieve gMSA passwords. Ensure that only necessary systems and accounts have this capability. Regularly rotate the root KDS key and consider isolating critical gMSAs by assigning them to specific, well-protected OUs.
- **Monitor** : Continuously monitor Windows event logs, specifically looking for **event ID 4662**, which indicates that someone is attempting to access gMSA attributes.
- **Detect** : Lucky for us, MDI will detect those activities if we have configured the **Advanced Auditing Policy** setting correctly. Detecting unauthorized access to the gMSA attributes early is crucial. If an attack is suspected, immediately rotate the KDS key and create new gMSAs to invalidate the compromised credentials.

Now that we've developed a strong understanding of how MDI action accounts work, let's shift our focus to how this knowledge can be practically applied. In the next section, we'll explore real-world scenarios and use cases that highlight the true power and versatility of MDI action accounts.

Real-world scenarios and use cases

With your MDI action accounts configured and secured, the next critical step is to understand how they can be leveraged in real-world situations. In this section, we'll delve into various scenarios where these accounts play a vital role in automating responses to threats.

Automated threat response – leveraging action accounts for quick reactions

When it comes to responding to security incidents, speed and precision are paramount. MDI action accounts allow you to automate responses to various detected threats, ensuring that immediate actions are taken to neutralize risks before they escalate. For example, if MDI detects suspicious activity such as multiple failed login attempts or the use of compromised credentials, the MDI action account can automatically disable the user account, preventing further access and mitigating the threat in real time.

Additionally, Microsoft Defender XDR enhances your organization's ability to handle active attacks through its **Automatic Attack Disruption** feature. This capability automatically responds to specific high-impact threats, such as ransomware or human-operated attacks, by isolating compromised devices, disabling user accounts, and stopping malicious processes. While these actions are triggered

independently by Defender XDR, they work alongside MDI's capabilities to create a comprehensive, multi-layered defense strategy that responds rapidly and effectively to emerging threats.

This synergy between automated responses in MDI and the broader disruption capabilities in Defender XDR ensures that your security operations can act swiftly across the entire kill chain, minimizing the damage caused by sophisticated attacks and enhancing overall security resilience.

As of the time of writing this book, MDI's capabilities for automated responses in these contexts include the following:

- Disabling user accounts
 - AD only account: Disable user account in local AD
 - Hybrid account: Disable user account in local AD and in Entra ID
 - Cloud account: Disable user account in Entra ID

These actions are targeted at your on-premises Active Directory environment. Therefore, it's crucial, as we have emphasized before, to ensure that your MDI environment is fully configured and operational. For Automatic Attack Disruption to function effectively in conjunction with MDI, the following prerequisites must be met:

- **Advanced auditing on domain controllers**: Proper auditing ensures that all necessary security events are logged and available for analysis by MDI and other Defender components. Test and verify your environment with, for example, the `DefenderForIdentity` MDI PowerShell module and the `Get-MDIConfigurationReport` cmdlet.
- **Validated action accounts with appropriate permissions**: MDI requires that action accounts have the necessary permissions to perform actions such as disabling users or resetting passwords. As we have learned, we can change how these accounts operate. This can be done under the `LocalSystem` account by default, but if customized with gMSAs, we must validate that they have the correct permissions at the correct scope (OU in Active Directory) to execute these critical tasks. To verify the gMSA, run the `Test-MDIDSA` cmdlet that is in the `DefenderForIdentity` PowerShell module.

IMPORTANT

It is best practice to use all Microsoft Defender products and make sure that they are deployed and configured correctly to use the full potential of the Attack Disruption feature in Microsoft Defender XDR. To learn more about the requirements of Attack Disruption feature for MDI, please visit:

<https://learn.microsoft.com/en-us/defender-xdr/configure-attack-disruption#microsoft-defender-for-identity-prerequisites>

By ensuring these configurations are in place, you enable MDI and Defender XDR to work together seamlessly, providing your organization with a robust and automated defense mechanism that is both proactive and reactive in addressing security threats.

I think we are ready to simulate some threats so we can try out the Attack Disruption feature.

Case study – detecting and responding to credential theft and lateral movement

Credential theft remains one of the most prevalent and dangerous attack vectors in cybersecurity. Attackers frequently exploit stolen credentials to gain unauthorized access to systems, which can lead to severe consequences such as data breaches or lateral movement within the network. In this consolidated scenario, we will demonstrate how MDI can detect when a user's credentials have been compromised – whether through a phishing attack, malware, or other means – and respond automatically to neutralize the threat.

When MDI detects a compromised user account, the configured MDI action account can immediately take critical steps to mitigate the threat. This includes disabling the compromised account and triggering a mandatory password reset. Additionally, these actions generate relevant security events, such as **event ID 4725** for account disablement, and alert the security team to ensure a quick and coordinated response. By automating this process, the risk of the attacker using the stolen credentials to access additional systems or move laterally within the network is significantly reduced.

In this case study, we will simulate a credential theft attack in a lab environment, demonstrating how MDI can effectively detect the compromise and utilize automated responses to contain and mitigate the threat before it can escalate further.

Step-by-step guide to simulating a credential theft attack

The objective is to simulate a credential theft scenario using the popular tool Mimikatz to extract passwords and trigger a detection in MDI. Make sure that you have fulfilled the prerequisites before moving on with this simulation.

NOTE

If **Tamper protection** is enabled, you will not be able to turn off Defender through the command prompt or PowerShell. You can, however, still create exclusions. For more guidance on managing and configuring **Tamper protection**, use the following link: <https://learn.microsoft.com/en-us/defender-endpoint/prevent-changes-to-security-settings-with-tamper-protection#how-do-i-configure-or-manage-tamper-protection>.

The prerequisites are as follows:

- Ensure that MDI is deployed and configured on your domain controllers.
- Have a test user account with domain administrator membership ready for the simulation. This account will be the target for credential theft.
- Disable Defender (see the commands later in this step-by-step guide).
- Install Mimikatz on a controlled machine within the lab environment (never use this in a production environment as it's a powerful tool often used by attackers).

Simulating credential theft with Mimikatz

Follow these steps to simulate the case study:

1. With your newly created test account, log in to any of the VMs that you have domain joined.
2. Start Windows PowerShell as an administrator.
3. Run the following commands:

```
# Disable real-time monitoring
Set-MpPreference -DisableRealtimeMonitoring $true
# Disable scanning for downloaded files or attachments
Set-MpPreference -DisableIOAVProtection $true
# Disable behavior monitoring
Set-MPPreference -DisableBehaviorMonitoring $true
# Make exclusion for a certain folder
Add-MpPreference -ExclusionPath "C:\Temp"
# Disable cloud detection
Set-MPPreference -DisableBlockAtFirstSeen $true
# Disable scanning of .pst and other email formats
Set-MPPreference -DisableEmailScanning $true
# Disable script scanning during malware scans
Set-MPPreference -DisableScriptScanning $true
# Exclude files by extension
Set-MpPreference -ExclusionExtension "ps1"
```

4. Ensure you are in a safe, isolated lab environment. Download Mimikatz from its official repository on GitHub from your VM to, for example, **C:\Temp** (which is set as an exclusion).
5. Extract the **.zip** file.
6. Open a command prompt or Windows PowerShell with administrative privileges.
7. Run Mimikatz by navigating to its directory and typing **mimikatz.exe**.
8. Execute the following commands to extract credentials from memory:

```
privilege::debug
sekurlsa::logonpasswords
```

If you get the **ERROR_kuhl_m_sekurlsa_acquireLSA** error message, then try the following commands:

```
!processprotect /process:lsass.exe /remove
privilege::debug
sekurlsa::logonpasswords
```

9. Mimikatz will display plaintext passwords, NTLM hashes, and other credential information from the current session.
10. After some time, you will find that your user was kicked out of the RDP session. Now, log in to the same VM with another account.
11. Open **Local Security Policy** (go to the Start menu and search for **Local Security Policy**) and navigate to **Local Policies > User Rights Assignment**. You'll notice that the account has been completely removed from the environment, as reflected in the settings shown here (see *Figure 8.6*):
 - Deny access to this computer from the network
 - Deny log on through Remote Desktop Services

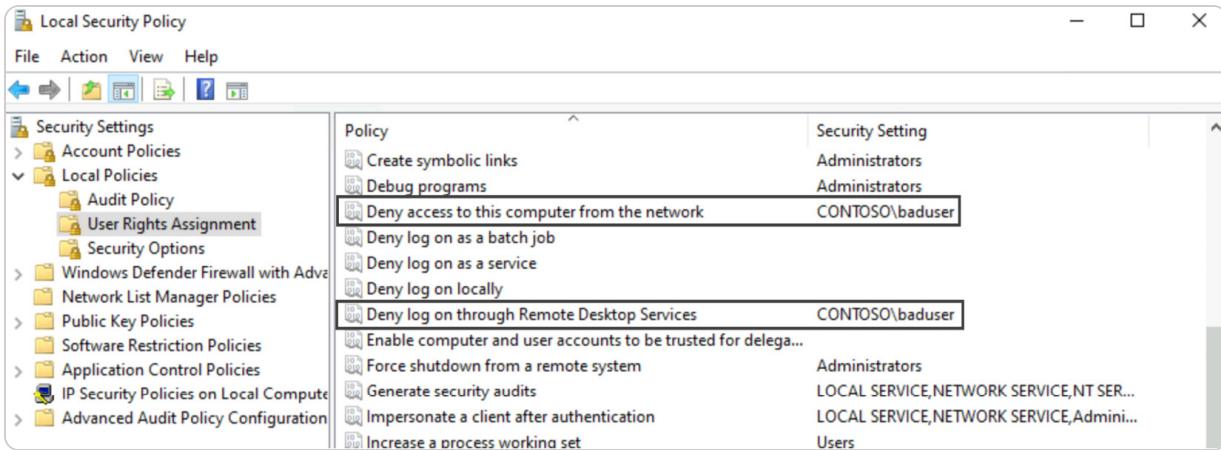


Figure 8.6 – Local Security Policy

12. In the Defender XDR portal, you will see a new incident with the **Attack Disruption** tag. Take some time to investigate the incident and the attack story (see *Figure 8.7*).



Hands-on keyboard attack was launched from a compromised account (attack disruption)

■■■ High | ● Active

Lateral Movement

Attack Disruption



Manage incident



Run playbook



Activity log

...

RELATED THREATS

Technique Profile: On-premises credential theft (Threat Overview)

176 impacted assets

[View threat analytics report](#)

Figure 8.7 – Incident with Attack Disruption tag

13. In the Defender XDR portal, navigate to the Action center to view all actions taken across devices, users, emails, and more.

Follow this path: [Investigation & response](#) | [Actions and submissions](#) | [Action Center](#).

As Mimikatz runs, Microsoft Defender XDR will detect some suspicious activity. Data and logs are correlated from the different Defender products (EDR, MDI, and so on), and alerts are linked to the bigger incident. In the coming section, we will see how we can make our own type of attack disruption with [custom detection rules](#).

Operational efficiency – how action accounts streamline security processes

In modern security operations, the ability to automate routine tasks is essential for maintaining an efficient and resilient security posture. MDI action accounts, when combined with the capabilities of Microsoft Defender XDR, allow you to automate key security actions, such as disabling compromised accounts, forcing password resets, and isolating devices. This automation not only accelerates response times but also ensures consistency and reduces the risk of human error.

To fully leverage the power of MDI action accounts and Defender XDR, you can create custom detection rules that trigger automated responses based on specific security events. For example, you might create a rule that automatically disables a user account and forces a password reset if suspicious activity, such as multiple failed login attempts, is detected.

Step-by-step guide to creating a custom detection rule

To create a custom detection that disables an on-premises account in Active Directory when we have a failed login event, follow these steps:

1. Log in to security.microsoft.com.
2. Navigate to **Investigation & response > Hunting > Advanced hunting**.
3. In the query field, type the following KQL query that identifies a failed log-on for our test user:

```
IdentityLogonEvents  
| where AccountName contains "baduser"  
| where ActionType == "LogonFailed"
```

This query detects the `baduser` user with a failed log-on to any device on the network. After defining the detection criteria, configure the automated actions.

4. Just above the query window, you will see a button labeled **Manage rules**. Click on that and then on **Create custom detection**.
5. In the **Alert details** window, make sure that you fill in the required fields (the following is just an example of values):
 - **Detection name** : For example, `Disable baduser`
 - **Frequency** : `Continuous (NRT)`
 - **Alert title** : For example, `Baduser was disabled in Active Directory`
 - **Severity** : For example, `Medium`
 - **Category** : For example, `Credential access`
 - **Description** : For example, `Looked at failed logon attempts for baduser`
6. In the **Impacted entities** window, choose **User** and then choose the column that contains the user ID (for example, `AccountObjectId`). Click on **Next**.
7. In the **Actions** window, and under **Users**, mark **Disable user**. Under **Choose user**, make sure that **AccountsId** is selected. Then, click on **Next**.
8. In the **Summary** window, make sure that you have configured the detection rule as you want it, then click on **Submit** to save the detection rule.

9. Let it take some time before you try to log in with this account and with the wrong password to see whether the account gets disabled in Active Directory.

By implementing these custom detection rules with automated responses, you can significantly enhance the efficiency and effectiveness of your security operations. Automating critical threat responses not only ensures swift and consistent action but also frees your team to focus on more strategic tasks. If you're eager to dive deeper into the world of security automation and build a truly resilient defense strategy, make sure to join us in the next chapter. But before that, let's summarize this chapter.

Summary

In this chapter, we explored the powerful capabilities of MDI action accounts and how they can be strategically used to enhance security operations within an organization. Starting with a detailed overview of how to configure and secure action accounts, we emphasized the importance of setting up these accounts correctly to ensure they are both functional and secure. Proper configuration and permission management are key to minimizing the attack surface and maximizing the effectiveness of automated responses.

We then delved into real-world scenarios, demonstrating how MDI action accounts can be applied to detect and respond to common security threats such as credential theft and lateral movement. By simulating these attacks in a lab environment, we illustrated how MDI's automated actions, such as disabling compromised accounts and forcing password resets, can significantly reduce the risk of further exploitation.

In the section on operational efficiency, we discussed how these automated actions not only accelerate response times but also ensure consistency in applying security policies, reducing the burden on security teams, and improving overall operational effectiveness. By automating routine tasks, organizations can better allocate their resources, focusing on strategic initiatives and complex threat investigations.

In the next chapter, we will dive even deeper into the world of automation, and more specifically on SOAR and the threat detection and response framework.

9

Building a Resilient Identity Threat Detection and Response Framework

In this chapter, we delve into the essential components of building a resilient **Identity Threat Detection and Response** (ITDR) framework using **Microsoft Defender for Identity** (MDI). As identity-based attacks become increasingly sophisticated, it's imperative to adopt proactive strategies that anticipate and mitigate threats before they impact your organization.

We begin by exploring how to design proactive **threat-hunting strategies** with MDI, using **Kusto Query Language** (KQL) to detect early indicators of compromise. You'll learn how to craft targeted queries that uncover hidden threats, enabling you to stay one step ahead of adversaries.

Next, we'll discuss how to elevate your **ITDR posture** – your organization's overall readiness and ability to detect and respond to identity-based threats – through continuous improvement. By integrating insights from incidents and aligning with best practices, you'll refine your defenses to adapt to the ever-changing threat landscape.

Finally, we'll cover disaster recovery and incident response, preparing you for the inevitable challenges of security breaches. We'll introduce resources such as the **Microsoft Incident Response Ninja Hub** to enhance your incident response capabilities and discuss how to develop effective strategies for swift recovery.

By the end of this chapter, you'll be equipped with the knowledge and tools to build a resilient ITDR framework that not only defends against identity threats but also prepares your organization for rapid response and recovery.

In this chapter, we'll cover the following main topics:

- Designing proactive threat-hunting strategies with MDI
- Elevating your ITDR posture – continuous improvement with MDI
- Disaster recovery and incident response – preparing for the inevitable

Technical requirements

Completion of [Chapter 2](#) and [Chapter 3](#) is a prerequisite for this section, and as a reminder, you also require the following:

- Microsoft tenant.
- Microsoft subscription that includes MDI, such as Microsoft 365 E5 or Microsoft 365 E3 + E5 Security.
- Basic Microsoft 365 knowledge.

- Active Directory knowledge.
- A virtual or physical server environment with Active Directory installed and configured. Optionally, you could have Active Directory Federation Services and Active Directory Certificate Services installed and configured.
- Basic PowerShell knowledge, and Windows PowerShell 5.1 or PowerShell 7.4 or later.

All the code examples for this chapter can be found on GitHub at

<https://github.com/PacktPublishing/Microsoft-Defender-for-Identity-in-Depth/tree/main/Chapter09>

Designing proactive threat-hunting strategies with MDI

In today's cybersecurity landscape, waiting for threats to knock on your door is no longer an option. **Proactive threat hunting** is about taking the initiative to seek out potential dangers before they become incidents. With MDI and all the other Defender tools, you have a powerful ally to anticipate and detect malicious activities targeting your identity infrastructure. In this section, we'll explore how to design effective threat-hunting strategies that use MDI's capabilities, enabling you to stay one step ahead of adversaries.

Understanding the threat-hunting methodology

Effective threat hunting is a thoughtful and methodical process rather than a random search for anomalies. At its core, it involves forming hypotheses about potential security threats based on available intelligence and then systematically investigating those hypotheses to confirm or refute them.

The journey begins with gathering information about the current threat landscape. This includes staying informed about the latest *attack techniques*, *vulnerabilities*, and *adversary behaviors* that could impact your organization. With this knowledge, you can formulate educated assumptions – or hypotheses – about where threats might exist within your environment.

For instance, you might hypothesize that attackers are attempting to exploit weak authentication protocols to gain unauthorized access to privileged accounts. This hypothesis provides a clear focus for your investigation.

Next comes the data collection and analysis phase. Using tools such as MDI, you delve into relevant data sources to find evidence supporting or disproving your hypothesis. This requires a keen understanding of normal versus abnormal behaviors within your network, as well as the ability to interpret subtle indicators that may signal malicious activity.

If your analysis uncovers signs of a threat, you proceed to the investigation phase, where you assess the scope and impact of the potential compromise. This might involve tracing an attacker's movements

through the network or examining changes made to critical systems.

Finally, you take action to mitigate the threat, such as isolating affected systems, updating security controls, or notifying relevant stakeholders. The insights gained throughout this process are invaluable for refining your threat-hunting strategies and improving your organization's overall security posture. And not actively working with all of the **Secure Score recommendations** within **Defender XDR** could potentially leave critical vulnerabilities unaddressed, exposing your organization to unnecessary risks and making it easier for attackers to exploit weaknesses in your defenses.

By starting to work on a well-defined methodology – whether through *structured* or *unstructured threat hunting* – you will gain valuable insights and practical experience, increasing the likelihood of detecting and neutralizing threats before they cause harm. Structured hunting follows a systematic approach based on predefined hypotheses and procedures, while unstructured hunting relies on the intuition and expertise of the analyst to explore anomalies. Both methods complement each other and contribute to a more robust security strategy.

At the time of threat hunting, it is important to develop a proactive threat-hunting mindset. While tools and technology are vital in threat hunting, the *human element* is equally crucial. Cultivating a proactive mindset within your security team means fostering curiosity and skepticism. Encourage your analysts to question anomalies, no matter how minor they may seem, and to delve deeper into unexplained events.

Collaboration is key to successful threat hunting. Create an environment where team members regularly share their findings and insights. Organize threat-hunting sessions where hypotheses can be tested and unusual patterns can be explored collectively. By promoting continuous learning and open communication, you enhance your team's ability to anticipate and counter threats.

Remember, proactive threat hunting is an ongoing journey, not a one-time project. Stay up to date with the latest threat intelligence, adapt your strategies as new attack methods emerge, and always be prepared to adjust your approach. With a proactive mindset and the robust capabilities of MDI, you'll be well-equipped to defend your organization's identity infrastructure against even the most advanced threats.

The importance of logging and accurate detection

Accurate and comprehensive **logging** is the backbone of successful *detection capabilities*, *threat hunting*, and *incident response*. Without the right logs capturing the necessary information, detecting and analyzing malicious activities becomes exceedingly difficult. Monitoring the ingestion of logs and not just the information in the log is as crucial as building detection queries.

Firstly, ensure that all critical systems, applications, and network devices are configured to *generate logs* that capture essential security events. This includes authentication attempts, access to sensitive resources, changes to configurations, and other activities that could indicate a security incident.

The *quality of your logs* matters as much as their availability. Logs should contain sufficient detail to support thorough analysis. This includes timestamps, source and destination information, user identifiers, and detailed descriptions of events. *Accurate time synchronization* across systems is crucial for correlating events effectively.

When it comes to detection, it's important to fine-tune your systems to reduce false positives and negatives. Overly sensitive configurations can overwhelm your team with alerts, leading to alert fatigue, while insufficient sensitivity might cause you to miss critical threats. *Regularly reviewing and adjusting your detection rules* based on evolving threats and organizational changes ensures that your detection capabilities remain effective.

Finally, having a well-defined *remediation process* is essential. Once a threat is detected, swift and coordinated action is required to contain and eradicate it. This involves clear communication channels, predefined roles and responsibilities, and documented procedures for common incident types.

By prioritizing proper logging, accurate detection, and efficient remediation processes, you lay a solid foundation for successful threat hunting and overall security operations.

Developing security use cases

Security use case development, also known as **use case modeling** or **use case engineering**, is a structured process that helps you define what threats to look for and how to detect them. By creating specific use cases, you tailor your threat-hunting efforts to address the most pressing risks to your organization.

For example, imagine a situation where an attacker attempts to gain access to confidential data by compromising a privileged account. This might involve using stolen credentials to impersonate a domain administrator, allowing unauthorized access to sensitive information. To prepare for this, you need to outline the steps of how such an attack could occur, identify what indicators to look for, and plan your response accordingly.

Here's a streamlined approach for building effective security use cases:

1. **Identify key assets and risks** : Determine which assets are most valuable, such as sensitive customer data, proprietary intellectual property, or critical infrastructure. Consider the specific risks these assets face, including unauthorized access, data breaches, ransomware attacks, insider threats, and service disruptions. Understanding these risks helps you focus on the most likely attack vectors and prioritize defenses around the assets that could cause the greatest harm if compromised.

2. **Establish detection criteria** : For each threat scenario, identify the data source and define what indicators would suggest that such an event is occurring. This could involve unusual login times, access from unfamiliar locations, or anomalies in user behavior.
3. **Leverage MDI capabilities** : Map these detection criteria to MDI's features. Utilize behavioral analytics, anomaly detection, and custom alerting to monitor these specific indicators.
4. **Document and prioritize use cases** : Create a repository of your security use cases, documenting the details and prioritizing them based on potential impact and likelihood. Even simple documentation, such as a PowerPoint slide, is better than nothing.
5. **Review and update use cases** : Regularly revisit and update your use cases to reflect changes in your organization's environment, evolving threat landscapes, and lessons learned from past incidents. This ensures your threat-hunting efforts remain effective and relevant.

By thoroughly developing security use cases, you ensure that your threat-hunting activities are focused and effective. This process also facilitates better communication within your team, as everyone operates from a shared understanding of the threats and detection methods. However, with over 3,000 different operations logged in the **Microsoft 365 suite**, figuring out which ones are most relevant for your organization can be quite a challenge.

That's where **Microsoft Incident Response guides** are invaluable. They highlight the key artifacts that **Microsoft Incident Response teams** hunt for and use daily to provide evidence of threat actor activity within customer environments. Think of these operations as pieces of a story – by concentrating on them, you can reconstruct an attack chain from beginning to end. This not only simplifies the daunting task of triaging and analyzing data in Microsoft 365 but also makes your threat-hunting efforts more efficient and impactful.

For example, monitoring *login events* such as successful sign-ins, failed attempts, reports of fraud without action taken, instances where users are blocked for **multi-factor authentication (MFA)** due to fraud reports, and any suspicious activity can reveal unauthorized access attempts or compromised accounts.

In the realm of *user activity*, keeping an eye on modifications to user objects is essential. This includes actions such as adding new users, changing passwords, users registering their security information, administrators registering security info on behalf of users, bulk downloads of user or service principal information, users initiating password resets, and account enablement. These events might indicate potential insider threats or external attackers manipulating user accounts to gain broader access.

Monitoring changes in the **identity plane** is also crucial. Actions (related to Microsoft 365) such as configuring federation settings on a domain, adding unverified domains, verifying domains, updating domain authentication methods, and other domain-related modifications could signal attempts to alter the foundational elements of your identity infrastructure.

Lastly, being vigilant about *impactful actions* – those that are destructive in nature – is essential. This includes removing members from roles, deleting conditional access policies, users or administrators deleting security information, deleting applications, setting accidental deletion thresholds, and bulk

deletion of users. Such activities can disrupt services, weaken security postures, and potentially indicate that an attacker is trying to cover their tracks or escalate their privileges.

MICROSOFT INCIDENT RESPONSE GUIDES

These new one-page guides from Microsoft Incident Response help security teams analyze cyber threat data in Microsoft 365 and Microsoft Entra: <https://www.microsoft.com/en-us/security/blog/2024/01/17/new-microsoft-incident-response-guides-help-security-teams-analyze-suspicious-activity/>.

With our newfound understanding of security use case development, we can now take the next step into hypothesis-driven hunting within our environment.

Leveraging behavioral analytics and MDI in hypothesis-driven hunting

In earlier chapters, we've explored the capabilities of MDI and how it enhances your organization's security posture. Now, we'll focus on applying proactive threat-hunting methodologies using MDI, specifically using the rich data available in its **hunting tables** – part of the advanced hunting schema that provides detailed information on events, devices, identities, and alerts. By combining a structured threat-hunting approach with MDI's powerful analytics, you can uncover hidden threats and strengthen your defenses.

Understanding hypothesis-driven threat hunting

Proactive threat hunting is a structured process that involves forming hypotheses about potential malicious activities and then investigating these hypotheses using available data. This method allows you to discover threats that might evade automated detection systems. For example, you might suspect that an attacker is trying to escalate privileges by modifying group memberships in **Active Directory (AD)**. With this scenario in mind, you'll follow a series of steps to validate or disprove your suspicion and uncover hidden threats:

1. **Gather data** : Begin by collecting data that relates to the suspected scenario. Query logs, event data, and telemetry from sources such as MDI (Defender XDR Advanced Hunting) or SIEM systems (for example, Microsoft Sentinel) to find signs of unauthorized changes to group memberships or other suspicious activities.
2. **Analyze data** : Use the scenario as a guide while examining the data for indicators that align with your hypothesis. Look for anomalies such as unexpected changes to group memberships, repeated access attempts from unfamiliar locations, or other patterns that might suggest an attempt to escalate privileges.
3. **Investigate findings** : If your analysis reveals potential threats, dig deeper into the scenario to determine the scope and impact. Investigate which accounts were used, the systems that were accessed, and any changes made to the AD environment. This will help you understand the full extent of the attack.

4. **Respond and mitigate** : Based on the scenario, take steps to mitigate the threat, such as reversing unauthorized changes to group memberships, isolating compromised systems, or resetting credentials and revoking active sessions. Quick action ensures that the attacker cannot maintain their access or cause further damage.
5. **Refine hypotheses** : Use what you've learned from the scenario and investigation to improve future hunting efforts. Adjust your hypotheses to account for similar tactics or adapt your detection criteria to catch such activities earlier next time.

With these steps outlined, you're now ready to begin formulating your own hypotheses for threat hunting. Before we dive in, let's revisit the hunting tables available in MDI in the following list:

- **IdentityDirectoryEvents** : Captures directory service-related activities, such as changes to user accounts and group memberships. You can also find system events on a domain controller here, such as activities from PowerShell and Task Scheduler.
- **IdentityInfo** : Contains information about identity objects from multiple identity sources.
- **IdentityLogonEvents** : Contains user authentication events, providing detailed information about user logins and logouts in AD, as well as authentication activities in Microsoft Online services monitored by Defender for Cloud Apps.
- **IdentityQueryEvents** : Contains data about queries made against AD.
- **ExposureGraphEdges** : Contains details about individual entities, such as users, devices, groups, and VMs. Each node represents a unique entity and provides attributes such as type and associated risk, which are essential for assessing exposure within the environment.
- **ExposureGraphNode** : Represents connections between entities in the graph, showing relationships between devices, users, and VMs. This table is crucial for identifying how resources are linked, which can be used to map possible lateral movement paths within the network.

As you start considering hypotheses to explore within your environment, let's look at how you can practically apply these concepts. We'll examine some real-world examples that illustrate the effective use of MDI's hunting tables in proactive threat hunting.

Practical use – Following the breadcrumbs with MDI

To illustrate how these concepts come together in practice, let's explore two real-world examples of how you can use MDI to hunt down identity threats.

Example 1 – Detecting unauthorized group membership changes

Hypothesis: An attacker is attempting to escalate privileges by adding their account to high-privilege groups.

We can approach the problem using the following steps:

1. **Formulate the hypothesis** : Attackers often try to gain elevated access by adding themselves or compromised accounts to privileged groups such as *Domain Admins* or *Enterprise Admins*. We hypothesize that any unexpected additions to these groups could indicate malicious activity and should be investigated promptly.
2. **Relevant data source** : Utilize the **IdentityDirectoryEvents** table in MDI, which captures group membership changes and other directory events in AD.

3. **Craft the query** : Using KQL, construct a query to retrieve recent group membership changes and see which accounts were added to which groups (you can easily add a `where` operator before the `project` operator at the end and filter by group if you want, for example, `where GroupModified == " Domain Admins"` :

```
IdentityDirectoryEvents
| where Application == "Active Directory"
| where ActionType == "Group Membership changed"
| where DestinationDeviceName != ""
| extend ToGroup = tostring(parse_json(AdditionalFields).["TO.GROUP"])
| extend FromGroup = tostring(parse_json(AdditionalFields).["FROM.GROUP"])
| extend Action = iff(isempty(ToGroup), "Remove", "Add")
| extend GroupModified = iff(isempty(ToGroup), FromGroup, ToGroup)
| extend Target_Group = tostring(parse_json(AdditionalFields)
["TARGET_OBJECT.GROUP"])
//| where GroupModified == "Domain Admins"
| project Timestamp, Action, GroupModified, Target_Account =
TargetAccountDisplayName, Target_UPN = TargetAccountUpn,
Target_Group, DC=DestinationDeviceName, Actor=AccountName,
ActorDomain=AccountDomain, AdditionalFields
```

Next, the following steps can be used to analyze the results:

1. **Execute the query and analyze results** : Begin by identifying anomalies and look for group membership changes such as these:
 - That were performed by accounts that do not typically make such changes
 - That occurred outside of normal administrative hours
 - That involve users who should not have privileged access

2. **Investigate suspicious findings** : For any group membership changes that seem unusual, do the following:

- **Validate legitimacy** :
 - Cross-reference with change management records to verify if the changes were authorized
 - Consult system administrators to confirm whether the additions are expected
- **Assess account activity** :
 - Review recent activities performed by the account that made the change.
 - Look for other anomalies, such as unexpected logins or access to sensitive resources.
 - Review any events where users were added to these groups. Verify if the changes were authorized.

3. **Cross-reference with IdentityLogonEvents** : Verify the logon events of the actor to see if their account has been compromised:

```
IdentityLogonEvents
| where AccountName contains "<username>"
| where Application == "Active Directory"
| summarize
TotalCount=count(),FirstSeen=min(Timestamp),LastSeen=max(Timestamp),SuccessCount=
countif(ActionType=="LogonSuccess"),ListOfSuccessfulDevices=make_set_if(DeviceName,
ActionType=="LogonSuccess"),FailureCount=countif(ActionType=="LogonFailed"),List
ofFailedDevices=make_set_if(DeviceName, ActionType=="LogonFailure") by
AccountName,DeviceName,LogonType
```

Analyze logon patterns for anomalies, such as logins from unfamiliar devices or locations.

4. **Respond and mitigate** : If unauthorized changes are detected, do the following:

- **Revert unauthorized changes** : Remove the unauthorized members from the privileged groups immediately
- **Secure compromised accounts** : Reset passwords and review permissions for any accounts involved
- **Conduct forensic analysis** :
 - Investigate the extent of the compromise
 - Look for additional indicators of compromise on affected systems
- **Update security measures** :
 - Implement monitoring for group membership changes as part of your standard security operations
 - Enhance access controls and review administrative privileges

5. **Document lessons learned** : This includes updating your threat-hunting procedures with insights gained and sharing knowledge with your team to improve overall readiness.

Example 2 – Monitoring service creation events for persistent threats

Hypothesis : An attacker is using unauthorized service creation on critical systems, such as domain controllers, to establish persistence within our network.

We approach this problem utilizing the following steps:

1. **Formulate the hypothesis** : Based on the understanding that service creation is a known method for attackers to maintain access, we hypothesize that any new services created on *domain controllers* or other sensitive systems could indicate malicious activity. Since these systems are typically stable and changes are infrequent, unexpected service creation should be investigated promptly.
2. **Gather relevant data sources** : Utilize the **IdentityDirectoryEvents** table in MDI, which captures a variety of directory change events in AD, including service creation events on machines protected by MDI.
3. **Craft the Query** : Construct a KQL query to retrieve recent service creation events:

```
IdentityDirectoryEvents
| where ActionType == "Service creation"
| project Timestamp, Application, AccountName, AdditionalFields.ServiceName
```

This query filters events where the `ActionType` is "Service creation" and projects relevant fields for analysis.

4. **Execute the query and analyze the results** : Execute the query to retrieve recent service creation events. Analyze the results to identify anomalies in these events, such as the following:
 - Unusual or nonsensical names
 - Created by accounts that do not typically perform such actions
 - Created on systems where service creation is rare or strictly controlled

5. **Investigate suspicious findings** : For any service creation events that seem out of the ordinary, do the following:

- **Validate legitimacy** : Cross-reference with change management records to verify if the service creation was authorized. Consult system administrators to confirm whether the service is expected.

- **Assess account activity** : Review recent activities performed by the account that created the service. Look for other anomalies, such as unexpected logins or access to sensitive resources.

6. **Respond and mitigate** : If you determine that the service creation is unauthorized or malicious, do the following:

- **Remove the malicious service** : Stop and disable the service immediately.
- **Isolate affected systems** : Disconnect compromised systems from the network to prevent further spread.
- **Secure compromised accounts** : Reset passwords and review permissions for any accounts involved.
- **Conduct a forensic analysis** : Investigate the extent of the compromise and look for additional indicators of compromise on the affected systems.
- **Update security measures** : Implement monitoring for service creation events as part of your standard security operations. Enhance endpoint protection to prevent unauthorized changes.

7. **Document lessons learned** : Update your threat-hunting procedures with insights gained. Share knowledge with your team to improve overall readiness.

With hypothesis-driven threat hunting and making use of MDI's capabilities, you can proactively detect and mitigate threats such as unauthorized group membership changes and service creations used by attackers to escalate privileges or establish persistence. To strengthen your defenses even further, it's essential to embrace continuous improvement. In the next section, we'll explore how to elevate your ITDR posture through ongoing refinement and optimization.

Elevating your ITDR posture – Continuous improvement with MDI

We all know that the threat landscape evolves very quickly, and so must your defenses. Proactively detecting threats is crucial, but it's equally important to continually refine and strengthen your ITDR strategies. By embracing a mindset of continuous improvement, you can stay ahead of adversaries and boost your organization's resilience against all types of attacks. In this section, we'll explore how to elevate your ITDR posture using MDI, drawing on lessons learned from real-world incidents and focusing on identity-hardening practices.

Learning from total identity compromise incidents

Total identity compromise represents one of the most severe threats an organization can face. When attackers gain full control over identity systems such as AD and **Entra ID**, they can access virtually any resource within the organization, disrupt operations, and exfiltrate sensitive data.

Understanding the attacker's tactics often follows a systematic approach to achieve total identity/domain compromise:

- **Initial access** : Gaining a foothold through methods such as phishing, exploiting vulnerabilities, or using stolen credentials.

- **Privilege escalation** : Elevating access rights by exploiting misconfigurations, abusing Group Policy, weak passwords, or unpatched systems (imagine other Tier 0 asset compromise).
- **Lateral movement** : Moving through the network to access additional systems and accounts, often using techniques such as **Pass-the-Hash** or **Pass-the-Ticket**.
- **Persistence** : Establishing long-term access by creating backdoor accounts, installing malware, or modifying authentication mechanisms.
- **Domain dominance** : Achieving control over AD or Entra ID, allowing unrestricted access to resources and the ability to manipulate security controls.

Common vulnerabilities that attackers exploit include the following:

- **Weak or compromised credentials** : Attackers exploit accounts with weak passwords or use credentials exposed in other data breaches.
- **Lack of MFA** : Without MFA, compromised credentials can be directly used to access critical systems.
- **Unpatched systems and legacy protocols** : Vulnerabilities in outdated systems or the use of legacy protocols such as **NTLM** and **LDAP** can be exploited.
- **Overprivileged accounts** : Excessive permissions granted to users or service accounts increase the risk if those accounts are compromised.
- **Misconfigurations** : Improper security settings, such as unrestricted delegation or vulnerable **Group Policy Objects (GPOs)**, provide attackers with opportunities to escalate privileges.

The key takeaway from total identity compromise incidents is the necessity for proactive defense measures. Organizations cannot afford to wait for an attack to occur; they must anticipate potential threats and strengthen their defenses accordingly.

By integrating *continuous improvement* into your security strategy, you can enhance your organization's resilience against such threats in various ways:

- **Regular assessments** : Continuously evaluate your identity security posture using MDI insights to stay ahead of evolving threats
- **Updating security policies** : Refine your security policies and procedures based on lessons learned to prevent repeat incidents
- **Training and awareness** : Educate your team about common attack vectors and the importance of adhering to security best practices

With a proactive approach and commitment to continuous improvement, you lay the groundwork for stronger defenses. Let's now explore how implementing identity-hardening strategies can further protect your organization.

Implementing identity-hardening strategies

To strengthen your ITDR posture, focus on identity hardening – a proactive approach to securing your identity infrastructure against attacks. The **Secure Score page** within the **Defender XDR portal** provides an excellent summary of risky account flags within your environment. This tool analyzes

your configuration and assigns a score based on the security posture of your accounts and settings. For each identified risk, it provides a detailed breakdown of affected accounts, along with step-by-step remediation guidance, helping you to address vulnerabilities directly from the portal.

In addition to using Secure Score, other specialized tools can also play a crucial role in identifying and addressing account vulnerabilities:

- **BloodHound** : This tool maps out AD relationships and permissions, identifying potential attack paths that an adversary might exploit. For example, it can pinpoint which user accounts have privileges that could be abused for lateral movement or privilege escalation, making it easier to target hardening efforts where they are most needed.
- **PingCastle** : A security assessment tool focused on AD, PingCastle performs in-depth audits to detect configuration weaknesses and compliance issues. It provides reports on outdated protocols, potential risks from legacy systems, and recommendations for improving security. For instance, it can highlight accounts that use weak encryption methods or that are configured with overly broad permissions.
- **Purple Knight** : Purple Knight is another powerful tool for assessing AD security. Developed by Semperis, it combines hundreds of checks to identify misconfigurations, privilege escalation paths, and potential attack vectors. Purple Knight's assessments cover numerous AD security risks, including risky delegation rights and privilege account misconfigurations. Each vulnerability is assigned a score and remediation guidance, helping organizations strengthen their defenses and prioritize identity-hardening tasks.

You can incorporate these tools into your identity-hardening efforts by using Secure Score for ongoing monitoring and prioritization and BloodHound, PingCastle, and Purple Knight for deeper assessments and remediation planning. This multi-tool approach ensures both high-level visibility and detailed analysis, providing a comprehensive view of potential vulnerabilities in your identity infrastructure.

Microsoft Secure Score

Overview Recommended actions History Metrics & trends

Export

38 items

Search

Filter

Filters: Product: Defender for Identity X

Rank	Recommended action	Score impact	Points achieved
1	Unsafe permissions on the DnsAdmins group	+0.62%	0/8
2	Resolve unsecure domain configurations	+0.38%	0/5
3	Disable Print spooler service on domain controllers	+0.38%	0/5
4	Protect and manage local admin passwords with Microsoft LAPS	+0.38%	0/5
5	Edit misconfigured certificate templates ACL (ESC4)	+0.38%	0/5
6	Edit misconfigured enrollment agent certificate template (ESC3)	+0.38%	0/5

Figure 9.1 – Secure Score recommended actions list

For a deeper understanding of how to proactively assess and improve your identity security posture, it's essential to keep up with the latest guidelines and recommendations.

READ MORE

Here, you will find the current list of proactive identity security posture assessments: <https://learn.microsoft.com/en-us/defender-for-identity/security-assessment>.

Effective identity hardening not only mitigates current risks but also builds a stronger foundation for long-term resilience. The following key strategies are essential for reducing your organization's exposure to identity-based threats:

- **Enforce MFA** : Ensure MFA is enabled for all users, particularly those with administrative access, to create an extra layer of security against unauthorized logins.
- **Reduce privileged access** : Regularly audit privileged accounts and adopt the principle of least privilege to ensure users only have the access necessary for their roles.
- **Secure service accounts** : Strengthen the security of service accounts with unique, strong passwords, and avoid interactive logins. Where possible, replace them with managed identities.
- **Implement Conditional Access policies** : Set up policies that require devices to be compliant or block access from risky locations, minimizing exposure to unauthorized access.

- **Harden AD infrastructure** : Secure your domain controllers and critical Tier 0 assets by limiting access, applying patches promptly, and using **privileged access workstations (PAWs)** for sensitive tasks.

While best practices are crucial, learning from real-world incidents further enhances your ability to defend against evolving threats. The following lessons from past identity compromises offer practical insights to strengthen your security posture:

- **Strengthen identity systems with MFA** : (Yes, you have seen MFA multiple times, configure it!) Many identity compromises could have been prevented with MFA, especially for privileged accounts. Attackers often exploit weak authentication practices, highlighting the need for strong verification methods.
- **Patch management is non-negotiable** : Unpatched systems are frequent entry points for attackers. High-profile incidents have shown that timely patching could mitigate a significant portion of identity-based attacks.
- **Limit exposure to legacy protocols** : Numerous breaches have taken advantage of outdated protocols such as NTLM or LDAP. Transitioning to modern authentication methods has become an essential step in preventing credential theft.
- **The cost of overprivileged accounts** : Overprivileged accounts make lateral movement and escalation easier for attackers. Lessons from past incidents reinforce the importance of continuously reviewing access permissions.
- **Monitoring and detection are key** : Incident response teams often find that early detection could have limited the damage of identity compromises. Tools such as MDI provide the visibility needed to catch anomalous activities before they escalate. Remember, collection of logs is not detection; make use of the logs.
- **Secure administrative practices** : Use dedicated, hardened workstations for administrative tasks to reduce the risk of credential theft from compromised devices.
- **Prepare for the worst with an incident response plan** : Post-breach analyses consistently reveal that having an up-to-date and rehearsed incident response plan can significantly reduce recovery times and minimize damage.

Now that we've explored identity-hardening strategies and real-world recovery scenarios, it's time to prepare for the unexpected. In the following section, we will focus on establishing a disaster recovery plan that ensures your identity infrastructure can quickly bounce back from a compromise, minimizing downtime and restoring security.

Disaster recovery and incident response – preparing for the inevitable

Cybersecurity incidents are no longer a matter of *if* but *when*. No matter how strong your defenses are, threats can and will find ways to challenge them. That's why it's essential to have a solid disaster recovery and incident response plan in place before any incident occurs.

Starting an incident response correctly means knowing what to do, when to do it, and who should be involved. It's not just about fixing the immediate problem; it's also about preserving important evidence and understanding any legal or regulatory steps you need to take.

Enterprise networks are complex, which makes defending them against determined attackers more challenging. Navigating the incident response process can feel like finding your way through a

labyrinth. Unfortunately, studies have shown that only about a quarter of organizations have an incident response plan that is consistently applied. Without one, you risk spending more time and money recovering from an incident than necessary.

Having a well-thought-out incident response plan can make the difference between quickly containing a threat and dealing with widespread business disruption. It helps your team act swiftly and efficiently, minimizing damage and downtime.

Establishing an incident response plan

Creating an effective incident response plan involves focusing on both the people and the processes that will be engaged during an incident. Here are key aspects to consider:

- **Define clear roles and responsibilities** : Assigning specific roles ensures that everyone knows their duties during an incident, reducing confusion and delays. An effective incident response team might include the following roles:
 - **Governance lead** : Typically, a senior executive such as a CISO or CIO who is responsible for operational oversight and maintaining visibility of risks and impacts to the business
 - **Incident controller** : Manages the operational aspects, coordinating all workstreams to understand and contain the threat
 - **Investigation lead** : Oversees the forensic investigation to understand the scope and nature of the compromise
 - **Infrastructure lead** : Focuses on threat containment and reducing risks posed by the attack
 - **Communications lead** : Manages internal and external communications to control messaging and keep stakeholders informed
 - **Regulatory lead** : Ensures compliance with legal obligations and manages interactions with regulatory bodies
- **Develop a structured response framework** : Adopting established methodologies provides a solid foundation for your incident response plan. The **National Institute of Standards and Technology (NIST)** provides a well-regarded framework that includes phases such as *preparation* , *detection* , *containment* , *eradication* , *recovery* , and *post-incident activities* . Tailoring this framework to your organization's specific needs helps in framing a tactical response plan, especially if you find yourself in a reactive position without a comprehensive plan already in place.
- **Set up secure communication channels** : Effective communication is vital during an incident. Setting up secure, out-of-band communication channels ensures that coordination efforts remain confidential. Threat actors may monitor standard communication lines, so using dedicated, secure channels helps maintain the integrity of your response and prevents the attacker from anticipating your actions.
- **Plan for team well-being and efficiency** : Incident response can be stressful and demanding. Implementing shift planning and resource allocation helps prevent burnout among team members. Ensuring that workloads are manageable and that staff have the support they need maintains high performance throughout the response.
- **Understand legal and regulatory obligations** : Engaging with legal counsel early in the process is crucial. Being aware of any laws or regulations that apply to your industry – especially concerning data breaches and reporting requirements – helps your organization navigate the incident without additional complications. Legal experts can guide you on mandatory reporting to authorities and help manage interactions with regulatory bodies.

- **Focus on documentation and communication** : Keeping detailed records of actions taken and decisions made is essential. Clear documentation supports effective communication across all teams and provides a single source of truth about the incident. Regular situation reports can keep key stakeholders informed and reduce the need for ad hoc updates, which can be time-consuming and disruptive.
- **Plan for continuous improvement** : After resolving an incident, conducting a thorough review helps identify lessons learned. This *post-incident analysis* is vital for refining your incident response plan and strengthening your organization's resilience against future threats. Incorporating these insights ensures that your team becomes more adept at handling incidents over time.
- **Build relationships with external partners** : Establishing contact with external entities such as law enforcement, cybersecurity experts, and key vendors before an incident occurs can be highly beneficial. *Early engagement* allows quicker assistance and support when you need it most, whether that's help with forensic analysis, threat intelligence, or fulfilling regulatory requirements.
- **Implement proactive monitoring and detection** : Utilize tools and processes that enable early detection of potential threats. *Proactive monitoring* can help identify suspicious activities before they escalate into serious incidents. Deploying **endpoint detection and response (EDR)** solutions and centralizing logs can enhance visibility across your network.
- **Consider the human element** : Recognize that people are at the core of your incident response. Ensuring that roles are well defined and that communication is clear helps prevent misunderstandings. Supporting your team through training and providing the necessary resources contributes to a more effective response.

If you haven't already, now is the time to start drafting your own incident response plan. Remember, in today's world, it's not a question of *if* an identity-based attack will occur, but *when*. With the right plan in place, you can be prepared to act swiftly when the inevitable happens. To try to be more proactive, we will now look at automated responses.

Automating responses to identity-based incidents with SOAR

As identity-based threats become more sophisticated, the ability to automate and orchestrate responses is crucial to minimizing damage and speeding up resolution. By integrating a **Security Orchestration, Automation, and Response (SOAR)** framework into your identity incident response process, you can reduce the manual workload and allow security teams to focus on more strategic tasks. Start collaborating with the cloud platform team to establish a workflow that uses the full potential of Azure services, allowing you to build an end-to-end solution for *automating responses*, *data enrichment*, and *orchestration*.

One of the key ways to automate responses is by using **Azure Logic Apps** together with **Azure Durable Functions**, triggered by incidents from Microsoft Defender XDR. This integration provides flexibility for creating complex workflows that automate threat mitigation, investigation, and data collection from various sources. With the recent support (introduced in mid-August 2024) for **PowerShell** within Azure Logic Apps, the possibilities for integrating custom scripts are almost limitless. This new capability empowers teams to execute advanced automation, such as dynamically modifying security groups, resetting passwords, or querying external threat intelligence databases, all from within Logic

Apps workflows. The ability to use PowerShell opens additional paths for automating highly tailored responses, enabling teams to quickly adapt to new attack methods without needing to build complex custom code from scratch.

In our scenario, we will call the **Durable Function** from the **Logic App** with **PowerShell** support. I hope in the future that we will get more support regarding the **Graph API** change events through **Azure Event Grid** – but that's another story; let's focus on what we can do today. In *Figure 9.2*, you can see a high-level architecture that we will use to enrich our incident with information:

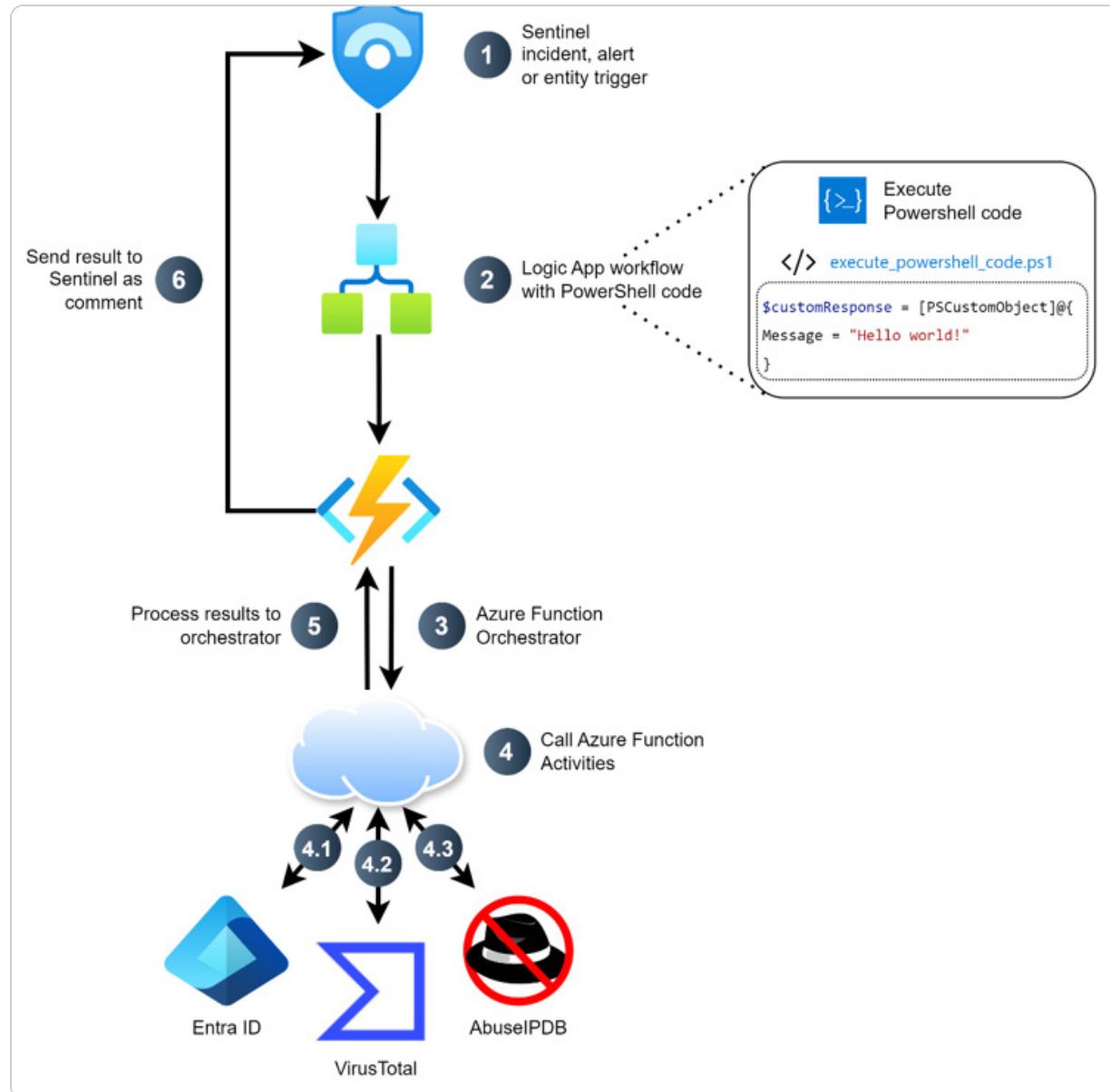


Figure 9.2 – Architecture SOAR

READ MORE

Discover how to integrate PowerShell scripts directly into Azure Logic Apps. This new feature enables powerful automation capabilities, making it easier to run custom scripts. Learn how to get started here: <https://learn.microsoft.com/en-us/azure/logic-apps/add-run-powershell-scripts>.

When an identity-related incident is detected by Microsoft Defender XDR, a Logic App can be automatically or manually triggered to initiate an orchestrated response. This automation can streamline the initial response by gathering and enriching the necessary information, reducing the time needed to investigate incidents manually.

In this context, the Logic App will trigger an **Azure Durable Function** to execute a series of automated tasks that enrich the incident with additional data and context for the analyst to review. At the time of writing this book, the PowerShell support in Logic Apps is still new and is only supported in the Standard model. The workflow will be a trigger action of **Microsoft Sentinel incident > Parse JSON data operator > Execute PowerShell Code**.



**Microsoft Sentinel
incident**



ParseJSON

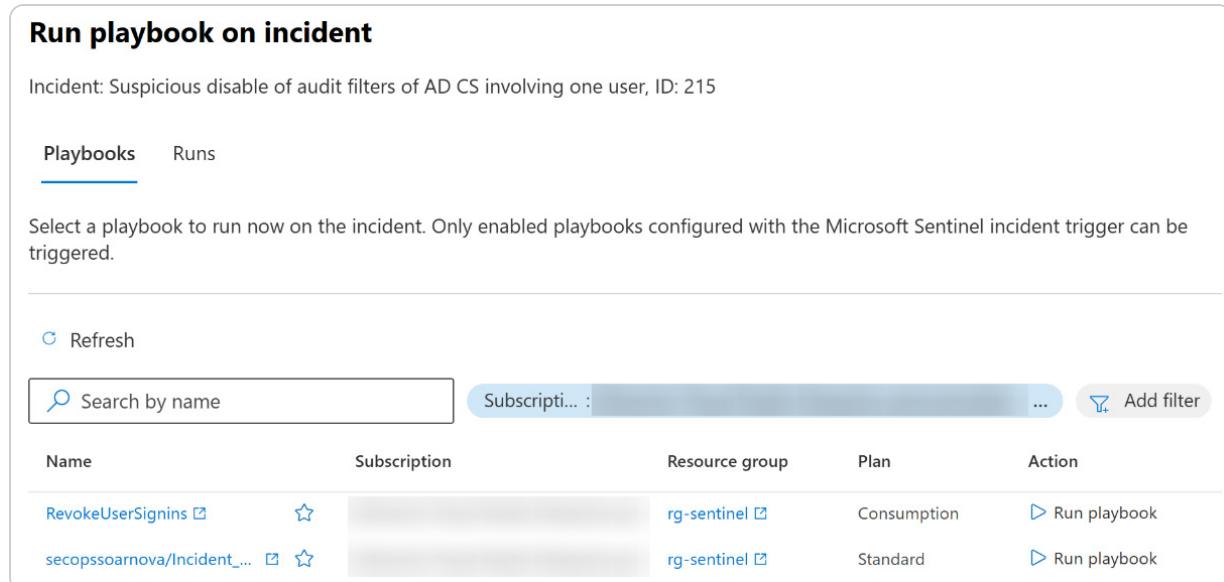


**Execute Powershell
Code**



Figure 9.3 – Azure Logic Apps standard workflow

To manually trigger a playbook from the Defender XDR portal, go to one of the incidents, then click on **Run playbook** (the button is in the top right, close to the title of the incident). You are then presented with some playbooks if you have configured everything correctly (see *Figure 9.4*) within the Logic App, and that means that you need (as of that moment) to have the incident trigger and not the entity or alert trigger within the Logic App.



The screenshot shows the 'Run playbook on incident' interface. At the top, it displays the incident details: 'Suspicious disable of audit filters of AD CS involving one user, ID: 215'. Below this, there are two tabs: 'Playbooks' (which is selected) and 'Runs'. A note below the tabs says, 'Select a playbook to run now on the incident. Only enabled playbooks configured with the Microsoft Sentinel incident trigger can be triggered.' There is a 'Refresh' button and a search bar labeled 'Search by name'. To the right of the search bar are buttons for 'Subscript... :', '...', and 'Add filter'. A table lists the available playbooks:

Name	Subscription	Resource group	Plan	Action
RevokeUserSignins	☆	rg-sentinel	Consumption	▷ Run playbook
secopssoarnova/Incident_...	☆	rg-sentinel	Standard	▷ Run playbook

Figure 9.4 – Run playbook on incident

MICROSOFT SENTINEL INCIDENT TRIGGER

Only enabled playbooks configured with the Microsoft Sentinel incident trigger can be triggered from incidents within the Defender XDR portal.

We will, in the coming section, start configuring the Logic App with PowerShell support and Durable Functions orchestrator and activities.

Orchestrating information gathering with Azure Durable Functions

By using **Azure Durable Functions**, you can orchestrate a complex workflow that involves collecting information from multiple sources. The **fan-out/fan-in** pattern of **Durable Functions** allows the parallel execution of tasks, which speeds up data collection and enrichment processes. Once all tasks are completed, the function can fan in and consolidate the enriched data, preparing it to be added as a comment to the incident. For identity-based incidents, this orchestration is vital for the following:

- **Enriching with threat intelligence** : Query external sources such as **AbuseIPDB** , a community-driven database that tracks malicious IP addresses reported for activities such as hacking, spamming, and attacks. This helps analysts quickly assess if an IP

address involved in an incident has a history of suspicious behavior, indicating it might be part of a larger attack.

- **Checking Users' Latest Sign-ins and Locations** : Entra ID sign-in logs can be queried to gather the user's recent login attempts, including login times and geographic locations. This helps identify if the account has been accessed from unusual locations, which could be an indicator of account compromise.
- **Device compliance status** : Check the compliance status of the device associated with the incident. This might involve querying Intune to see if the device meets the organization's security standards, ensuring that only compliant devices are allowed access to sensitive data.
- **IP geolocation and reputation** : Fetch the geolocation and reputation of the IP addresses involved, cross-referencing multiple threat intelligence databases to provide context about the legitimacy of the activity.
- **Correlation with other incidents** : Using Defender XDR, correlate the current incident with other incidents or alerts involving the same user, device, or IP address. This gives a broader view of whether the incident is part of a coordinated attack.

Let's see how the fan-in/fan-out pattern would look in a PowerShell-based Durable Function orchestrator.

Orchestrator function

The **orchestrator** controls the workflow. It triggers multiple tasks in parallel (fan-out) and waits for them all to complete (fan-in). In PowerShell, this can be done using the `Start-DurableTask` cmdlet to run activities in parallel.

Here's a simple example:

```
function Run-Orchestrator {
    param([OrchestrationContext] $context)
    # Fan-out: Trigger parallel tasks
    $tasks = @()
    $tasks += Invoke-DurableActivity -FunctionName 'Query-AbuseIPDB' -Input $context
    $tasks += Invoke-DurableActivity -FunctionName 'Check-SignInLogs' -Input $context
    $tasks += Invoke-DurableActivity -FunctionName 'Check-DeviceCompliance' -Input
    $context
    # Fan-in: Wait for all tasks to complete
    $results = Wait-DurableTask -Task $tasks
    # Process the aggregated results
    $finalResult = @{
        AbuseIPDB = $results[0]
        SignInLogs = $results[1]
        DeviceCompliance = $results[2]
    }
    # Return consolidated results
    return $finalResult
}
```

The steps involved are as follows:

1. **Fan-out** : The orchestrator triggers multiple activities in parallel (e.g., querying AbuseIPDB, checking sign-in logs, and checking device compliance). These tasks run concurrently, reducing the overall execution time. `Invoke-DurableActivity` starts each activity. Each activity performs an asynchronous action, such as querying an external API or querying logs.
2. **Fan-in** : Next, the orchestrator uses `Wait-DurableTask` to wait for all the tasks to be completed. Once they're finished, it consolidates the results into a final output.

3. **Return the results** : The orchestrator aggregates the results from all the parallel tasks and returns them as a single object, which can then be added as a comment to a Defender XDR incident.

With a new understanding of how the fan-out/fan-in pattern works to speed up data collection and analysis, we can now turn our attention to the next component of this process: **activity functions**. These functions are the building blocks that execute each of the parallel tasks in the orchestration. Let's explore how they work and how they contribute to efficient information gathering.

Activity functions

Activity functions are individual tasks within an orchestrated workflow designed to perform a specific operation, such as querying databases, making API calls, or analyzing data. These functions help automate processes such as gathering threat intelligence or analyzing user sign-in patterns.

For example, an activity function could query AbuseIPDB to enrich MDI alerts with additional information about an IP address involved in a potential incident:

```
function Invoke-Query-AbuseIPDB {
    param([ActivityContext] $context)
    # Query AbuseIPDB for IP reputation
    $ip = $context.InputIpAddress
    $result = Invoke-RestMethod -Uri "https://api.abuseipdb.com/api/v2/check?
    ipAddress=$ip" -Headers @{
        "Key" = "API_KEY"
        "Accept" = "application/json"
    }
    return ($result.content | ConvertFrom-Json).data
}
```

Similarly, you can create your own activity functions, such as `Check-SignInLogs` to gather the latest sign-ins for a specific user and `Check-DeviceCompliance` to check the status of a device compliance status.

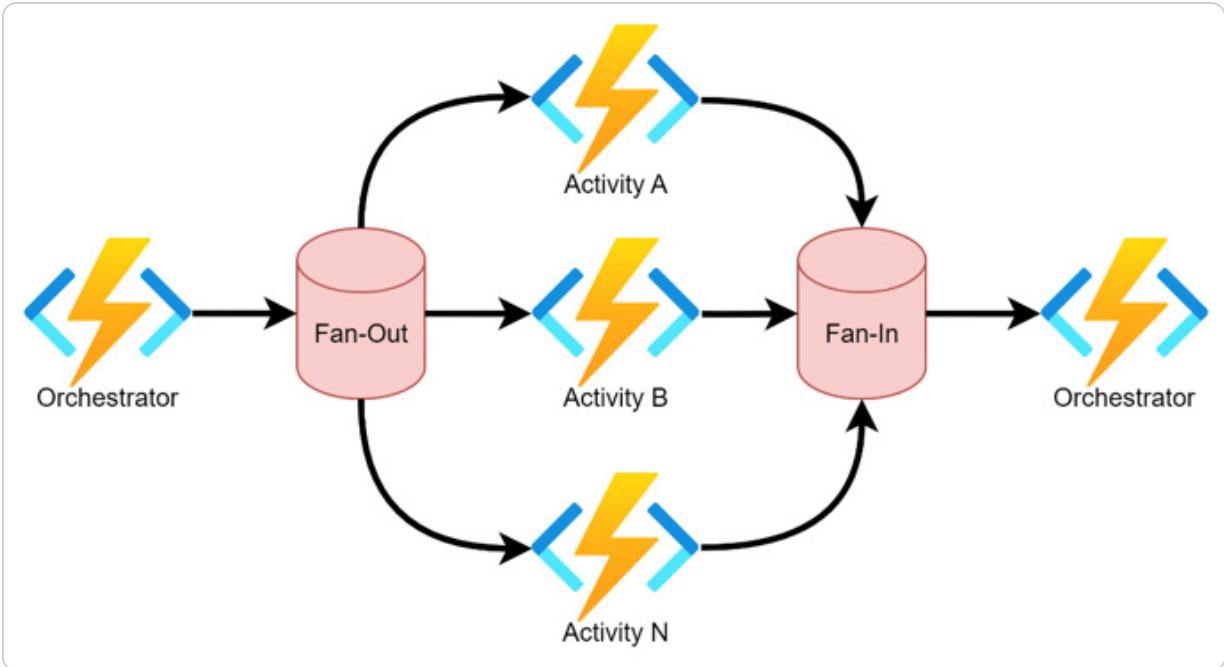


Figure 9.5 – Fan out/fan in pattern

With this type of solution and framework, you can easily start building more modularly and use **Git** (via Azure DevOps or GitHub) for code versioning, testing, validation, and deployment through pipelines, making it a true SOAR as code. We will now look at an event that I hope you never will face – disaster recovery.

Disaster recovery for identity systems

In the event of an identity compromise involving on-premises AD, it's crucial to follow a structured recovery process that ensures not only the restoration of services but also the eradication of any attacker footholds. This recovery must include securing privileged accounts, removing persistence mechanisms, and reinforcing security measures to prevent further exploitation. Let's look at the following guide as a starting point:

- Audit and remove unauthorized privileged group memberships :** Begin by auditing the membership of critical privileged groups such as **Domain Admins**, **Enterprise Admins**, and **Backup Operators**. Attackers often target these groups to maintain persistence. Ensure that only authorized users remain in these groups, removing any unnecessary or unauthorized members. This step should also include checking for other groups with delegated administrative privileges over critical systems, as attackers may have added accounts to these groups to maintain control.

NOTE

See Microsoft's Securing Privileged Access model to ensure privileged accounts are appropriately restricted:

<https://learn.microsoft.com/en-us/security/privileged-access-workstations/privileged-access-deployment>

2. **Reset passwords for all privileged accounts** : Resetting passwords for all privileged accounts is crucial for locking out any attackers who may have gained access to those accounts. This includes not only members of built-in privileged groups but also those in groups with delegated access to critical systems, such as server and workstation administrators.

NOTE

Ensure the new passwords meet your organization's complex password policies and consider applying MFA to all administrative accounts for added security.

3. **Reset the `krbtgt` account twice to invalidate Kerberos tickets** : One of the most critical steps in identity recovery is resetting the `krbtgt` account password twice. Attackers who have compromised AD can create **golden tickets** – forged Kerberos tickets that grant them unlimited access to the domain – using this account, which allows them to maintain access to the domain even after their other access points are removed. Resetting the `krbtgt` password twice effectively invalidates these tickets. This process involves the following steps:

- I. Use Microsoft's official script to reset the `krbtgt` password: [https://github.com/microsoft/New-KrbtgtKeys.ps1](https://github.com/microsoft/New-KrbtgtKeys.ps1/blob/master/New-KrbtgtKeys.ps1).
- II. After allowing enough time for replication, reset the password a second time with a different value to fully invalidate old Kerberos tickets.

4. **Remove attackers' persistence mechanisms** : Persistence mechanisms left behind by attackers must be thoroughly removed to prevent re-compromise. These mechanisms may include the following:

- Malicious services running with elevated privileges
- Unauthorized scheduled tasks
- Modifications to GPOs
- Changes to user accounts, passwords, or permissions

Using **Defender XDR and Event Viewer**, review logs for any signs of unauthorized changes or suspicious activity. Ensure all malicious entries are removed.

5. **Restart domain controllers to flush memory-resident malware** : Some malware may remain in memory even after persistence mechanisms are removed. A restart of all domain controllers will clear *memory-resident malware* that could otherwise continue running undetected. Schedule restarts after confirming that persistence mechanisms have been removed.
6. **Reset the DSRM password for all domain controllers** : The **Directory Services Restore Mode (DSRM)** password is a critical recovery credential that allows you to boot into AD's **recovery mode**. Ensuring that this password is unique and secure for each domain controller helps prevent attackers from exploiting it for further access. Follow these steps to reset the DSRM password on each domain controller:

- Log in to one of the domain controllers where you need to reset the DSRM password.
- Open **Command Prompt** as an administrator on the domain controller.
- In Command Prompt, type the following command and press *Enter*
- **ntdsutil**
- Once you are inside the **ntdsutil** utility, enter the following command to begin resetting the DSRM password

- **set dsrm password**
- After entering the **set dsrm password** mode, reset the password for the desired domain controller by running the following command, replacing <ServerName> with the name of the domain controller
- **reset password on server <ServerName>**
- You will be prompted to enter a new password for DSRM. Ensure the password is *complex* and *unique*, following your organization's password policy.
- Once the password has been reset, quit the **ntdsutil** utility by typing **q** and pressing *Enter*.
- Perform these steps on each domain controller in your environment to ensure all DSRM passwords are updated.

7. Validate the recovery and perform post-recovery audits : Once the recovery steps are completed, perform a comprehensive audit of the environment:

- *Review AD replication* to ensure that all changes have propagated correctly
- *Audit security logs and event logs* for any remaining signs of compromise or unusual activity
- *Ensure that all privileged accounts have had their passwords reset* and no unauthorized access remains

I really hope that you never need to do the previous section, but I cannot stress enough that you need to be prepared for it. The key to a faster, more effective recovery lies in training, practice, and continuous refinement of your disaster recovery processes. By investing time in simulations and refining procedures, your organization will be better equipped to respond efficiently and minimize the fallout from an identity compromise. With the right combination of people, processes, and technology, you can ensure that even in the face of adversity, recovery will be instantaneous, and security will be restored. Now, let's summarize this chapter.

Summary

In this chapter, we explored the critical components of building a resilient identity threat detection and response framework. Starting with the design of a robust incident response plan, we emphasized the importance of clear roles, processes, and communication to ensure a swift and coordinated response to identity-based incidents. We then delved into the power of automating incident response with SOAR capabilities, using Azure Logic Apps and Durable Functions to enrich identity incidents with actionable data from multiple sources, allowing faster and more efficient incident management.

Furthermore, we covered disaster recovery strategies specific to on-premises AD, outlining a structured approach to restoring identity services in the event of a compromise. This included resetting critical privileged accounts, removing persistence mechanisms, and safeguarding the environment from future threats. The key takeaway from this chapter is that preparation, training, and continuous improvement are vital for reducing recovery times and ensuring that security can be restored rapidly and effectively when the inevitable occurs.

By combining people, processes, and technology, organizations can fortify their defenses, respond to identity-based threats with confidence, and recover from potential compromises swiftly. Your identity infrastructure is the backbone of your security posture and, as we've discussed, having a comprehensive plan in place makes all the difference when an incident strikes.

In the next chapter, we will explore the most common troubleshooting scenarios you may encounter while working with MDI.

10

Navigating Challenges: MDI Troubleshooting and Optimization

Welcome to the final chapter of our journey with **Microsoft Defender for Identity**! Here, we dive into the exciting world of **troubleshooting** and **optimization**, turning challenges into opportunities for a stronger security setup. Whether you're facing common hiccups or looking to fine-tune your system for peak performance, this chapter is your go-to guide.

Think of this chapter as your trusty toolbox. I'll walk you through how to identify and fix typical MDI issues, resolve tricky configuration and connectivity problems, and enhance your system's performance to keep everything running smoothly. Plus, I'll show you how to manage those annoying false alarms, making your security alerts smarter and more reliable.

You'll learn how to detect and correct misconfigurations that might be holding your system back, tackle network connectivity issues to ensure seamless communication, and make sure the right event logs are being audited for comprehensive monitoring. Additionally, we'll explore the operational guide to give you inspiration to be on top of the daily to quarterly tasks for good MDI hygiene.

By the end of this chapter, you'll feel confident tackling any MDI challenge that comes your way.

In this chapter, we'll cover the following main topics:

- Diagnosing common MDI issues
- Configuration and connectivity fixes
- Resolving security alert misfires
- Operational guide

Let's turn every challenge into an opportunity for a better, more secure system!

Diagnosing common MDI issues

Let's face it – every system has its bumps in the road. With MDI, knowing how to find and fix these bumps quickly is key to keeping your security strong. In this section, I'll show you how to identify common MDI problems and get them sorted out fast.

Managing MDI means you might run into some unexpected issues now and then. Whether it's *weird behavior*, *slow performance*, or *missed threats*, being able to diagnose and fix these problems is essential. Here, I will walk you through the most common MDI issues and give you easy steps to find and fix them.

We'll start by looking at the signs that something's wrong with your MDI setup. Then, I'll show you how to use MDI's *built-in tools* and *logs* to dig deeper and find the root of the problem. You will also see some best practices to keep your settings in check and make sure everything is working together smoothly. By the end of this section, you'll be a pro at spotting and fixing common MDI issues, keeping your system reliable and secure.

Spotting the signs of trouble

In MDI, the beauty of the system lies in its simplicity – detection rules are entirely handled by Microsoft. This means you don't have to worry about fine-tuning detection rules yourself, as MDI automatically updates to stay on top of the latest threats. However, as with any system, things can occasionally go wrong. Even though the rules are out-of-the-box, knowing how to recognize when something isn't quite right is critical. In this section, we'll focus on identifying the subtle signs that might indicate deeper issues within your MDI deployment.

While you may not be able to see or modify the detection rules, you can still monitor how well they are working by keeping an eye out for these common warning signs:

- **Missed or delayed alerts** : Although Microsoft takes care of the detection rules, if MDI isn't generating alerts for activities you expect to be flagged, there may be an underlying issue. This could happen if MDI is not receiving the necessary telemetry data from your domain controllers or other sensor servers. Network problems or connectivity issues can cause gaps in data collection, leading to missed or delayed alerts.
- **Unusual activity in logs** : Sometimes, the trouble is hiding in plain sight. If you're seeing unusual patterns in your event logs – such as unexpected spikes in network traffic, failed login attempts, or strange access patterns – yet MDI hasn't triggered an alert, there might be a problem with data collection. While MDI should normally detect and flag suspicious behavior, issues such as incomplete logs or sensor disconnections can prevent it from recognizing a threat.
- **Performance lags** : If your MDI logs ingestion feels slow – taking longer than usual to process events or generate alerts – it's worth investigating further. Performance issues could point to bottlenecks, especially in larger environments with high traffic. Overloaded sensors, underutilized servers where the sensor is installed, or insufficient data processing can delay detections, leaving your organization vulnerable to threats that MDI hasn't had time to process.

While monitoring these common warning signs is important, certain key areas require special attention to ensure MDI is functioning smoothly. Let's take a closer look at where you should focus to prevent or quickly resolve any issues:

- **Defender XDR portal** : This is your control center for everything regarding your security with Defender products, including MDI. Regularly reviewing the portal for missing or incomplete alerts is critical. If you notice a gap in alerts – especially for activity that should have triggered one – it could be a sign that something's not working as it should. It might not always be a missed alert due to a system flaw; instead, it could be due to gaps in telemetry or connectivity issues. Your job is to spot the absence of expected alerts and investigate further.
- **Event logs and network connectivity** : MDI depends heavily on event logs and seamless connectivity between its sensors and your domain controllers. If MDI isn't receiving these logs, it's essentially blind. Ensuring that event logs are complete and being

processed without interruption is a top priority. Additionally, make sure your MDI sensors are always connected to the network so they can continuously monitor and analyze events.

- **Monitoring network traffic** : The flow of data between MDI sensors and the MDI cloud service must be steady. If you notice unusual drops in network traffic, interruptions in communication, or errors in data transmission, this could be the root of missed detections. For instance, if a sensor disconnects from the domain controller for even a short period, MDI might miss critical information needed to trigger an alert.

In the next section, we'll dive deeper into the tools MDI offers for diagnosing and troubleshooting these problems, so you can get to the root of the issue quickly and efficiently.

Using tools and logs to find problems

Once you've spotted the early signs of trouble in your MDI environment, the next step is to dig deeper. MDI provides a range of built-in tools that can help you pinpoint exactly where things are going wrong. From checking the health of your sensors to analyzing logs, these tools are essential for ensuring your security system is running smoothly. In this section, we'll explore the different diagnostic tools and logs you can use to track down problems in your MDI setup.

Microsoft Defender for Identity Health Center

Your first stop when investigating issues should be the **Microsoft Defender for Identity Health Center** (also known as the [Health Issues page](#)) within the Defender XDR portal. This dashboard gives you a *bird's-eye view* of how well your MDI sensors are functioning. If a sensor is disconnected or lagging, the necessary configuration is not applied, or it's not receiving event logs from the sensors installed in your environment, this dashboard will notify you. Regularly checking the health of your sensors is crucial, as any interruption can create gaps in your security coverage. In [Chapter 3](#), we covered how to set up custom alert functionality for the MDI sensor. *Figure 10.1* shows an example of a built-in health issue alert triggered by a disconnected sensor.

Sensor stopped communicating

■■■ Medium ● Open

 Close health issue  Suppress

Description

There has not been communication from the Sensor ADFS0.contoso.local for 9/9/24 1:23 PM - 9/22/24 9:16 PM. Last communication was on 9/9/24 1:23 PM.

Generated time

Sep 9, 2024 1:34 PM

Last Updated

Sep 9, 2024 1:34 PM

Recommendations

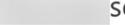
- Check that the Sensor service is up and running.
- Check the communication between the Sensor to  sensorapi.atp.azure.com:443.

Figure 10.1 – Sensor stopped communicating

When reviewing the **Health Issues** page, there are several key factors to monitor:

- Sensor connectivity is critical. Ensure that all your sensors are online and consistently communicating with the MDI cloud service. If one or more sensors aren't connected, it's important to troubleshoot the network connection and confirm that the sensor is properly configured.
- Processing delays can indicate performance issues. If you notice delays in sensor processing, this might mean that the sensor is struggling to handle the volume of data it's receiving. Misconfigurations or large amounts of data can cause these delays, which in turn affect the speed at which alerts are generated.
- Data collection is another crucial element. It's essential to verify that the sensors are collecting telemetry data from all monitored devices. Missing logs or incomplete event data can prevent MDI from detecting suspicious activity, potentially leaving your organization vulnerable.

The **Health Issues** page helps monitor overall sensor connectivity and performance, but sometimes, you need to dive deeper for a more thorough investigation. When performance issues persist or data isn't being collected as expected, it's time to explore the event logs. These logs offer a detailed breakdown of sensor operations and errors, giving you the insights needed to diagnose and resolve issues that may not be immediately visible from the dashboard.

Event logs – your diagnostic backbone

While the **Health Issues** page offers a high-level view, the real investigation often begins with **event logs**, which provide detailed records of sensor operations. These logs are critical for diagnosing issues in your MDI environment. The default location of these logs is `C:\Program Files\Azure Advanced Threat Protection Sensor\ version number\Logs`.

Start by reviewing `Microsoft.Tri.Sensor.log`, which tracks overall sensor activity, and `Microsoft.Tri.Sensor-Errors.log` for error-specific insights. These logs will help identify problems such as network issues or delays in data collection.

Additionally, `Microsoft.Tri.Sensor.Updater.log` and its error counterpart, `Microsoft.Tri.Sensor-Updater-Errors.log`, are key when troubleshooting sensor updates. The **updater logs** track the status of automatic sensor updates, while the error log highlights issues that occurred during the update process. These logs are particularly useful for health checks and diagnosing problems that happened at specific times in the past.

The main operational logs include the following:

- `Microsoft.Tri.Sensor.log` : Tracks sensor activity and overall operations
- `Microsoft.Tri.Sensor-Errors.log` : Focuses on errors within the sensor
- `Microsoft.Tri.Sensor.Updater.log` : Logs details of sensor update processes
- `Microsoft.Tri.Sensor.Updater-Errors.log` : Captures errors encountered during sensor updates

In addition to these operational logs, **deployment logs** are crucial for ensuring a smooth setup of the MDI sensors. These logs track the entire installation process, from deploying the sensor to installing its binaries. Typically, you'll find these logs in the `Temp` directory of the user who installed the sensor, `%USERPROFILE%\AppData\Local\Temp`, or in `c:\Windows\Temp` if deployed by a service. They are necessary for resolving installation-related issues and confirming that the sensor is deployed correctly.

Key deployment logs include the following:

- `Azure Advanced Threat Protection Microsoft.Tri.Sensor.Deployment.Deployer.log` : Logs the entire deployment process
- `Azure Advanced Threat Protection Sensor_YYYYMMDDHHMMSS.log` : Tracks steps during the sensor deployment phase
- `Azure Advanced Threat Protection Sensor_YYYYMMDDHHMMSS_001_MsiPackage.log` : Captures the installation of the sensor binaries

While reviewing logs can help identify underlying issues with sensor performance and update processes, another critical aspect of maintaining MDI involves ensuring that your sensor configuration and network connectivity are properly aligned. Any misconfigurations or blocked

connections can lead to missed detections and other operational challenges. Let's explore the next steps in troubleshooting and fine-tuning your MDI setup.

Configuration and connectivity fixes

Getting MDI configured correctly and making sure it stays connected is super important for keeping your security tight. If configurations are off or connections drop, you could leave gaps that hackers love to exploit, and we do need the automatic **Attack Disruption** feature. In this section, I will guide you through common setup mistakes and connectivity headaches, offering simple solutions to get everything back on track.

First, we'll look at the main configuration settings that MDI needs to work properly. Then, we'll tackle connectivity problems such as network drops, firewall blocks, and sensor issues, giving you straightforward ways to fix them.

Checking key configuration settings

To ensure MDI is running optimally, it's important to regularly review the main *configuration settings*. Misconfigurations can lead to missed detections, performance bottlenecks, or even sensor failures. In this section, we'll walk you through some of the most common errors you might encounter and provide clear steps to fix them, so you can keep your MDI deployment running smoothly.

Common errors and fixes

Before diving into specific troubleshooting fixes, it's important to recognize that a smooth MDI deployment relies heavily on correctly configured settings that we have been focusing on in the earlier chapters (see *Chapters 1 and 3* for installing and configuring the different MDI sensors).

Misconfigurations or system limitations can often lead to sensor issues, such as missed detections or performance bottlenecks. To help you stay ahead of these issues, I've outlined some of the most common errors you might encounter and how to resolve them quickly. Here is a list of issues and their solutions:

- **Installation wizard stuck for several minutes :**
 - **What it means :** If you try to install the MDI sensor through the wizard and want to use a proxy for the network communication but the wizard gets stuck, and the relevant services go from **starting** to **not started**, it could mean that it doesn't read the input correctly.
 - **How to fix it :** Make sure to fully uninstall the MDI sensor. For more information, see the section called *Removing a malfunctioning MDI sensor manually*. Then try to install the MDI sensor again through the command line using the following syntax:

```
".\Azure ATP Sensor Setup.exe" [/quiet] [/Help]
```

```
[ProxyUrl="http://proxy.domain.com"] [ProxyUserName="domain\proxyuser"]
[ProxyUserPassword="Password"]
```

- Group Managed Service Account (gMSA) credentials not retrieved :

- **What it means :** The sensor might fail to retrieve the gMSA password, preventing it from functioning properly.
- **How to fix it :** Ensure that the server has the necessary permissions to access the gMSA credentials. Verify that the gMSA account is properly configured and that the network connectivity required to retrieve the credentials is functioning without interruption. Also, make sure to have a security group where all of your MDI sensor servers (AD DS, AD CS, AD FS, and Entra Connect) will be a member. Run the following command in a PowerShell window within one of your domain controllers, replacing the name of the gMSA account that you have created:

```
Get-ADServiceAccount -Name <name of gMSA> -Properties
PrincipalsAllowedToRetrieveManagedPassword
```

In the output, you should see either the security group or the individual servers under the `PrincipalsAllowedToRetrieveManagedPassword` property (see *Figure 10.2*).

```
PS C:\> Get-ADServiceAccount -Identity MDIGMSA -Properties PrincipalsAllowedToRetrieveManagedPassword

DistinguishedName : CN=MDIGMSA,CN=Managed Service
                     Accounts,DC=contoso,DC=local
Enabled          : True
Name             : MDIGMSA
ObjectClass      : msDS-GroupManagedServiceAccount
ObjectGUID       : 00e75f6b-9cab-490b-a2c9-6c4a60f101ff
PrincipalsAllowedToRetrieveManagedPassword : {CN=Domain
                                             Controllers,CN=Users,DC=contoso,DC=local, CN=MDIGrou
                                             p,OU=Groups,OU=Contoso,DC=contoso,DC=local}
SamAccountName   : MDIGMSA$
SID              : S-1-5-21-2219569903-3511389863-1636676309-1106
UserPrincipalName : 
```

Figure 10.2 – Output from Get-ADServiceAccount

UPDATING KERBEROS TICKETS AFTER GROUP MEMBERSHIP CHANGES

If you have a security group for your MDI sensors and you add a new server to the group, ensure that you reboot the server or run the following command in an elevated command prompt:

```
klist -li 0x3e7 purge
```

This command forces the server to synchronize and obtain a new Kerberos ticket that includes its updated group membership.

- NIC teaming issues :

- **What it means :** If your sensor is installed on a machine configured with NIC teaming and using WinPcap drivers, you may encounter errors due to compatibility issues.
- **How to fix it :** MDI now supports **Npcap** instead of WinPcap. Download the latest MDI sensor version from the Defender XDR portal (see [Chapter 2](#)) and make sure to uninstall the old sensor with WinPcap via **Add/Remove Programs** or run the following command in an elevated prompt:

```
".\Azure ATP Sensor Setup.exe" /uninstall /quiet
```

- Local service runs the MDI service (Azure Advanced Threat Protection Sensor) on the server :

- **What it means :** This is by design from Microsoft. If you have configured the gMSA account and wonder why the service still uses the Local Service, it is used for outbound connections (SAMR and LDAP).
- **How to fix it :** There is nothing to fix. Just make sure that you follow the best practices by using a security group to gather your MDI sensor servers and that the gMSA account works on those servers.

With the new knowledge of some of these common configuration issues, you can significantly enhance the stability and performance of your MDI deployment. However, there may be cases where a sensor becomes unresponsive or cannot be uninstalled through standard methods. In such cases, manual removal is necessary to resolve the issue and allow for a clean reinstallation. In the next section, we will look at the steps to manually remove a malfunctioning MDI sensor.

Removing a malfunctioning MDI sensor manually

Occasionally, when working with MDI, you may encounter sensors that cannot be installed, uninstalled, or removed through standard methods. In such cases, manually removing the sensor becomes necessary to resolve the issue.

A practical approach involves using PowerShell to delete the sensor's services installed on the server, along with named pipes, registry entries, and associated files. This ensures a complete cleanup, allowing you to start fresh with a new sensor installation. This method is particularly useful for stubborn issues that other troubleshooting steps cannot resolve.

NAMED PIPES

MDI sensors use named pipes for critical inter-process communication between their components. The named pipes are specifically as follows:

```
\.\.\pipe\CPFATP_<aatpsensorPID>_v4.0.30319
\.\.\pipe\CPFATP_<aatpsensorUpdaterPID>_v4.0.30319
```

In these pipe names, <aatpsensorPID> / <aatpsensorUpdaterPID> represents the process ID of the MDI sensor or its updater service. These named pipes facilitate communication necessary for the sensor's operation and are important to be aware of when troubleshooting or performing manual cleanup tasks.

For a detailed walkthrough of the process, you can inspect this helpful guide by MVP Michael Morten Sonne, which thoroughly explains each step:

<https://blog.sonnes.cloud/microsoft-defender-for-identity-how-to-manually-remove-a-malfunctioning-sensor-that-can't-be-installed-uninstalled-or-removed/>

The PowerShell script he developed, which you can find on his page, is designed to perform a complete cleanup of the MDI sensor installation. Here's a detailed breakdown of its functionalities:

- **Disable, stop, and remove services :** The script identifies and manages the services associated with the MDI sensor, ensuring they are disabled, stopped, and removed from the system

- **Terminate running processes** : Any processes related to the MDI sensor are terminated to prevent conflicts during the cleanup
- **Remove named pipes** : The script identifies and removes any named pipes created by the MDI sensor
- **Uninstall NPCAP** : Uninstalls NPCAP, the packet capture driver used by the MDI sensor to capture and analyze network traffic for threat detection
- **Delete installation folders and files** : All folders and files associated with the MDI sensor installation are deleted
- **Clean up registry keys** : The script cleans the registry by removing keys related to the MDI sensor
- **Remove ATP certificates** : Any MDI (formerly Advanced Threat Protection) certificates installed by the MDI sensor are removed

This method gives you full control when the built-in uninstaller fails or if the sensor becomes stuck in an unresponsive state. Once removed, you can proceed with *reinstalling* a fresh MDI sensor without conflicts from traces of the previous installation.

With the malfunctioning sensor completely removed and a fresh installation in place, it's important to ensure that the new sensor operates smoothly within your network environment. The next critical step is to verify and troubleshoot any network connectivity issues that could affect the performance and reliability of your MDI deployment. Let's proceed to explore strategies for identifying and resolving network connectivity problems.

Network connectivity troubleshooting

Ensuring proper communication between MDI sensors and the MDI backend is essential for smooth deployment and operation. Here are troubleshooting tips and real-world scenarios focused on resolving connectivity and sensor issues.

Testing MDI sensor connectivity

If your MDI sensor isn't communicating with the backend, it's essential to test connectivity as early as possible and to start troubleshooting. You can use PowerShell to run diagnostic tests that simulate the sensor's communication with the MDI backend.

Using PowerShell

Following enlist the steps involved in testing using PowerShell:

1. Start **Windows PowerShell** or **PowerShell 7** on the MDI sensor server.
2. Run the following command to test the connection to the MDI backend:

```
Invoke-WebRequest -Uri https://<your-instance-name>.sensorapi.atp.azure.com/tri/sensor/api/ping
```

For Windows Server Core edition, make sure to use the

`-UseBasicParsing` parameter within the `Invoke-WebRequest` cmdlet:

```
Invoke-WebRequest -UseBasicParsing -Uri https://<your-instance-name>sensorapi.atp.azure.com/tri/sensor/api/ping
```

Replace `<your-instance-name>` with your actual MDI instance name.

WINDOWS SERVER CORE

The `-UseBasicParsing` parameter directs the cmdlet to handle HTML content without parsing it with the **Document Object Model (DOM)**. This is necessary for systems where Internet Explorer is not available, such as Windows Server installations using Server Core. Without DOM support, the cmdlet uses a simpler, more basic method to interpret HTML responses.

3. Next, it's time to understand the result:

- **Status Code 200:** Indicates that the sensor can communicate successfully with the MDI cloud service
- **Status Code 503:** For older MDI workspaces, a 503 Service Unavailable code is expected and confirms successful routing to the backend
- **Failure:** Suggests issues with network connectivity, proxy configurations, or firewall settings blocking the connection

Using the MDI PowerShell module

Before you begin, ensure that the **Defender for Identity PowerShell module** is installed on your server. This module provides the necessary cmdlets for testing and managing MDI sensor connectivity.

Once you've verified that the Defender for Identity PowerShell module is installed, you can test the sensor's connectivity using different approaches based on your requirements. Whether you want to verify the connection using the current server configuration or test it with specific settings, there are two main options available:

- **Option A - test using current server configuration :** To test connectivity using the server's existing settings, open PowerShell and run the following:

```
Test-MDISensorApiConnection
```

This command checks whether the sensor can successfully connect to the MDI cloud service using the current configuration, including any proxy settings.

- **Option B - test using specific settings :** If you want to test connectivity with settings that aren't currently applied to the server, use the following syntax:

```
$credential = Get-Credential
```

```
Test-MDISensorApiConnection -BypassConfiguration -SensorApiUrl 'https://<your-instance-name>sensorapi.atp.azure.com' -ProxyUrl 'https://your-proxy-server:port' -ProxyCredential $credential
```

Replace the placeholders as follows:

- `https://<your-instance-name>sensorapi.atp.azure.com` : Your actual sensor API URL, where `your-instance-name` is the name of your MDI workspace (observe that there's no dot between the instance name and the `sensorapi` part)

- `https://your-proxy-server:port` : The URL and port number of your proxy server
- `$credential` : A PowerShell credential object containing your proxy authentication details

Understanding the results from Options A and B : After running these tests, you'll receive one of two outcomes: `True` or `False`. These results reflect whether the sensor was able to successfully communicate with the MDI cloud service based on the configuration you're testing. The test simulates the actual communication process, validating whether the current network, proxy, and firewall configurations are correctly set up to allow the sensor to connect to the MDI service:

- **True** : Indicates that the sensor can communicate with the MDI cloud service using the specified settings
- **False** : Suggests issues with network connectivity, proxy configurations, or firewall settings blocking the connection

If your connectivity tests indicate a failure, it's possible that proxy settings are interfering with the sensor's ability to communicate with the MDI cloud service. Proxies can impact network traffic in various ways, especially if they are misconfigured or not properly accounted for in your sensor setup. In the following section, we'll delve into common proxy configuration issues and provide guidance on how to resolve them to restore proper communication between your MDI sensors and the cloud service.

Proxy configuration issues

A misconfigured proxy can easily block communication between MDI sensors and the cloud service. Proper proxy configuration is essential to ensure that MDI functions correctly. Here are some common scenarios to consider:

- **Pre-configured proxy settings** : Before diving into specific configurations, ensure your existing proxy settings allow communication between MDI sensors and necessary cloud services. Review the following aspects to prevent connectivity issues:
 - **Verify allowed URLs** : If your domain controllers are already using proxy settings, ensure that the URLs required for MDI communication are allowed through the proxy. Specifically, confirm that outbound HTTPS traffic on port **443** is permitted to the following MDI URLs:


```
*.atp.azure.com
crl.microsoft.com
ctldl.windowsupdate.com
www.microsoft.com/pkiops/*
www.microsoft.com/pki/*
```
 - **Check proxy authentication** : If your proxy requires authentication, make sure the MDI sensor is configured with the necessary credentials. MDI sensors support basic and NTLM proxy authentication.
 - **Proxy Bypass for Local Addresses** : Ensure that the proxy settings do not inadvertently bypass proxy usage for addresses that should be routed through it.
- **No Proxy Settings Configured** : When no proxy settings are pre-configured, but your network requires one for internet access, you'll need to manually set up a proxy for the MDI sensors. With the Defender for Identity PowerShell module, you can retrieve the current proxy settings using the following:

```
Get-MDISensorProxyConfiguration
```

If you need to configure or update the proxy settings, use the following command:

```
Set-MDISensorProxyConfiguration -ProxyUrl 'http://proxy.domain.com:8080'
```

This command updates the sensor's proxy settings to route the MDI traffic through your specified proxy server.

- **TinyProxy configuration issues :** When using TinyProxy as your proxy solution for MDI, it's important to ensure that the configuration is tailored to support the sensor's communication needs. Failure to do so can lead to connectivity issues between your sensors and the MDI cloud service. Here are the key configurations to verify and adjust:

- **Specify sensor IP addresses :** If you're using TinyProxy as your proxy solution, ensure that the IP address or IP range of the MDI sensor servers is correctly specified in the TinyProxy configuration to allow connections.
- **Adjust MaxClients value :** Verify that the **MaxClients** setting in the TinyProxy configuration is set high enough to handle the expected number of simultaneous connections from your sensors.
- **Allow required URLs :** Check the TinyProxy filter file to ensure that the necessary MDI URLs are included in the allowlist:
 - **Filter File Location :** You can find the TinyProxy filter file at `/etc/tinyproxy/filter`. This file controls which URLs are allowed to pass through the proxy.
 - **Required Entries :** Add the following entries to the filter file to ensure proper connectivity:

```
*.atp.azure.com  
crl.microsoft.com  
ctldl.windowsupdate.com  
www.microsoft.com/pkiops/*  
www.microsoft.com/pki/*
```

These entries allow the MDI sensors to communicate with the necessary MDI services.

By confirming that your proxy settings are correctly configured, you remove one of the common obstacles that can prevent MDI sensors from communicating with the cloud service. Nevertheless, even with proper proxy settings, other network factors can interfere with sensor connectivity. One such factor is SSL inspection, which can disrupt the operation of the MDI sensor's communication.

Let's explore how SSL inspection can cause issues and what steps you can take to address them.

SSL inspection causing issues

In environments with SSL inspection, it's common for MDI sensors to fail communication if SSL inspection isn't properly configured. MDI requires mutual authentication, which can be disrupted by SSL inspection if it interferes with the secure handshake process. Here's how you can identify and resolve potential SSL inspection problems:

- **Symptoms :** Sensors fail to connect to the MDI backend, and the deployment logs may display SSL-related errors.

- **Resolution** : Work with your network security team to ensure that SSL inspection is configured to allow mutual authentication. Disable SSL inspection for the MDI URLs to prevent issues.
- **Investigating logs for deployment and sensor issues** : If deployment or sensor communication fails, logs are the first place to look for root causes. Depending on the stage of failure, different logs provide valuable insights:
 - **Deployment logs** : If the sensor deployment is unsuccessful, check the logs located in either `%USERPROFILE%\AppData\Local\Temp` or `%temp%` for installation failures. These logs will often reveal issues such as incorrect proxy settings, SSL inspection problems, or firewall blocks.
 - **Sensor logs** : For sensors that are not communicating after installation, check the sensor logs located at `C:\Program Files\Azure Advanced Threat Protection Sensor\version number\Logs`

Common issues include changes to proxy configurations or firewall rules that block communication between the sensor and the backend.

Another network-related issue that can occur, especially in larger enterprises, is the **Network Name Resolution (NNR)**. We will now be heading to that section and learning how to troubleshoot it.

Troubleshooting NNR issues in MDI

One critical function of MDI is NNR, which resolves IP addresses to computer names, enabling accurate correlation of activities. Without proper NNR, alerts such as *suspected identity theft (Pass-the-Ticket)* or *DCSync attacks* may be flagged incorrectly, resulting in false positives.

In complex environments, issues with NNR can significantly impact the effectiveness of MDI. When NNR isn't functioning properly, it can lead to misidentified devices and inaccurate correlation of activities, which hampers threat detection and response. To proactively address these challenges, it's important to recognize the common symptoms that indicate NNR issues. These include the following:

- Repeated alerts for the same device (Pass-the-Ticket)
- Alerts triggered by a domain controller or DNS server without valid cause

To troubleshoot, ensure that the MDI sensors can connect to network devices over the primary protocols used by NNR:

- **TCP Port 135** for NTLM over RPC
- **UDP Port 137** for NetBIOS
- **TCP Port 3389** for RDP

Additionally, MDI relies on **reverse DNS lookup (UDP Port 53)** as a fallback when primary methods fail. Make sure your DNS environment is properly configured with reverse lookup zones, and that devices can dynamically register their **Pointer (PTR) records**.

If these connections fail, check firewall rules or subnet access restrictions.

In a big environment with many DNS servers and latency issues involving PTR records, NNR can face several challenges. These include the following:

- **Delayed or failed hostname resolution** : When DNS servers slow down or network configurations are misaligned, resolving IP addresses to hostnames becomes delayed or may fail altogether:
 - **High latency** : Slow responses from DNS servers can cause delays in resolving IP addresses
 - **Timeouts** : Extended latency may lead to query timeouts, resulting in unresolved IP addresses
 - **Impact on alerts** : Unresolved IPs may cause MDI to generate alerts with less context, making incident response more difficult
- **Increased DNS query load** : In large environments with numerous devices, the volume of DNS queries can spike significantly:
 - **High volume of queries** : Large environments generate more network traffic, leading to more DNS queries from MDI sensors
 - **DNS server overload** : Excessive queries can strain DNS servers, further increasing latency
- **Inaccurate or missing PTR records** : Maintaining accurate PTR records is essential for reliable IP-to-hostname resolution, and issues in this area can cause inconsistencies:
 - **PTR record misconfiguration** : In large networks, PTR records may be outdated or incorrectly configured
 - **No reverse mapping** : Without accurate PTR records, MDI cannot resolve IPs to hostnames effectively

For further debugging, you can use a custom **KQL query in Defender XDR Advanced Hunting** to identify sensors unable to connect over these critical ports. This will help pinpoint connectivity issues between sensors and network segments.

Using the `DeviceNetworkInfo` together with the `DeviceNetworkEvents` table, you can start troubleshooting the network blocks.

You can also view NNR-related health issues on the **MDI Health Issues page**. Any disruptions or low success rates will be flagged here, providing insight into potential network or sensor misconfigurations.

Addressing NNR-related health issues is a significant step toward ensuring accurate threat detection. However, even with NNR functioning optimally, you might still experience misfires in security alerts due to other underlying factors. Let's delve into strategies for resolving these security alert misfires to enhance the effectiveness of your MDI deployment.

Resolving security alert misfires

Security alerts are your first line of defense, but when they go off too often for the wrong reasons, it can be both frustrating and distracting. Frequent false alarms can undermine your trust in the security

system and divert attention from genuine threats. Fine-tuning your alerts in MDI is essential to ensure you're notified only about real security issues.

In this section, we'll explore why false positives happen and how to prevent them. You'll learn how to adjust alert thresholds, customize detection rules, and apply filtering techniques to minimize unnecessary alerts. By tailoring your MDI alert settings, you can maintain strong security oversight without feeling overwhelmed.

Understanding the root causes of false positives is the first step toward mitigating them. Legitimate activities that resemble malicious behavior – such as unusual user actions, automated processes, or network configuration changes – can trigger unnecessary alerts. Recognizing these patterns allows you to adjust your settings to better align with your organization's typical activities.

With a clear understanding of why false positives occur and the strategies to reduce them, it's time to put this knowledge into practice. In the next section, I'll guide you through adjusting your MDI alert settings to enhance accuracy and reduce unnecessary notifications.

Customizing detection rules and applying filtering techniques

To further reduce false positives and modify MDI to your environment, you can customize detection rules and apply filtering techniques. These methods allow for more granular control over alert generation, ensuring that your security team focuses on genuine threats.

See the following steps for adding an exclusion, such as device and or account.

How to create exclusions and whitelists

The following steps need to be implemented:

1. Navigate to **Settings > Identities > Exclusions by detection rule** in the Defender XDR portal.
2. Add users, devices, or IP addresses, known to generate false positives.
3. Select specific alerts to exclude these entities from.

How to implement suppression rules

I will start by saying that the following part is not recommended because it can create blind spots, but if you need to have a suppression rule for any business-related requirements, here's how to do it:

1. When viewing an alert, click **Tune alert**.
2. Define conditions such as specific entities or alert types.
3. Suppression rules prevent alerts from being generated based on your criteria.

Now that we can (with caution) create and implement some filtering, it's time to see how we can adjust the alert threshold for MDI.

Adjusting alert settings for better accuracy

To fine-tune *MDI alert thresholds*, you can customize alert behavior to match the specific needs of your environment, reducing false positives and optimizing detection. MDI allows you to adjust thresholds for individual alerts based on your organizational needs, ensuring that alerts trigger at appropriate levels without overwhelming your security team.

Before modifying alert thresholds, it's important to assess your current alerting patterns and identify which alerts are most prone to false positives. Engage with your security team to gather insights on recurring issues and determine the optimal sensitivity levels for different types of alerts.

Let's proceed to understand how you can adjust these thresholds within MDI.

How to adjust alert thresholds

In the Microsoft Defender XDR portal, go to **Settings > Identities > Adjust Alert Thresholds**. Here, you will find the list of all alerts where you can adjust the thresholds.

Choose the specific alert you want to fine-tune. By default, the threshold is set to **High**, which minimizes false positives by applying stricter detection criteria. You can lower the threshold to **Medium** or **Low** depending on your operational needs.

After selecting the appropriate threshold (**Medium** or **Low**), click **Apply Changes** to save your settings. You can also revert to the default settings if needed. Remember to document any changes made to the alert thresholds for future reference and include why you decided to change the threshold. It's also advisable to inform your security team about these adjustments so they can interpret alerts appropriately and adjust their monitoring strategies if necessary.

To help you make an informed decision about which threshold level to choose, here's a breakdown of each *alert level* and its implications:

- **High (Default)** : Ensures that only high-confidence alerts are triggered, reducing false positives but potentially missing lower-confidence threats
- **Medium** : Triggers more alerts by slightly loosening detection conditions, allowing you to capture more potential threats
- **Low** : Triggers the highest number of alerts; useful for comprehensive testing or environments where you want maximum coverage at the cost of potential false positives

You can also enable **Recommended Test Mode** to temporarily set all thresholds to **Low** for more aggressive testing of your security setup. When finished, simply turn off test mode to return to normal operation.

This fine-tuning capability is especially useful in dynamic environments where legitimate activities (such as VPN connections or NAT usage) might otherwise trigger false positives. For example, adjusting the **Suspected DCSync attack** detection (this alert indicates that an account is attempting to replicate Active Directory data by mimicking a domain controller) to a **lower threshold** helps monitor replication requests more aggressively, particularly in environments where credential misuse is a concern. However, it's important to strike a balance. Setting the threshold too low may result in a high number of false positives, overwhelming your security team with alerts and potentially causing genuine threats to be missed amid the noise. Since **High** is the default setting, it's essential to balance sensitivity with practicality to ensure your team can effectively manage and respond to alerts.

By carefully adjusting alert thresholds and understanding their impact, you can tailor your security monitoring to better fit your organization's needs. With the alert thresholds configured appropriately, let's now focus on establishing a structured approach to maintaining your MDI deployment.

Operational guide

Maintaining a secure and efficient MDI deployment requires regular monitoring and adjustments. By implementing a clear maintenance schedule, you can ensure that your MDI sensors stay operational, detect threats accurately, and maintain overall system performance. In the coming sections, you will see a breakdown of tasks categorized by daily, weekly, monthly, and quarterly cadences, along with ad-hoc tasks for addressing unexpected issues and to be at the top of your game.

Daily tasks

Daily maintenance and reviews are critical to ensure your MDI deployment is operating smoothly and staying ahead of potential threats. Here is a distinguished daily task guide, designed to prioritize incident triage, looking at high-risk users, system health checks, and proactive threat hunting:

- **Prioritize and triage incidents** : Start each day by addressing the most pressing security alerts. By focusing on severity and potential impact, you ensure that no critical threat goes unaddressed:
 - **What to do** : Start your day by reviewing all new incidents, sorting them based on severity. Focus on high-priority alerts such as credential theft attempts or lateral movement within your environment.
 - **Why it matters** : Addressing the most critical threats first ensures that potential security breaches are contained before they cause widespread damage.
- **Investigate high-risk users** : Some users may be flagged due to unusual or suspicious behavior, making them a higher priority for investigation. Delving into these cases early helps to mitigate risks before they escalate:
 - **What to do** : Identify users who have been flagged with a high investigation score. This score reflects potentially suspicious activities that warrant a closer look.

- **Why it matters :** High-risk users may indicate ongoing attacks or compromised accounts. Investigating them quickly can prevent further damage or lateral movement.
- **Fine-tune detection rules :** Keeping detection rules finely calibrated ensures you are catching true threats while avoiding unnecessary noise. Regular adjustments to these detection rules will help maintain an effective security posture:
 - **What to do :** Review any alerts that turned out to be false positives or benign true positives. Adjust detection rules accordingly to reduce noise and improve the accuracy of future alerts.
 - **Why it matters :** Regular tuning helps prevent alert fatigue by minimizing unnecessary notifications, so you can focus on real threats.
- **Review the Identity Threat Detection and Response (ITDR) dashboard :** The ITDR dashboard gives a comprehensive overview of identity-related threats. Regular reviews will keep you informed about ongoing issues or new risks, allowing you to take timely action:
 - **What to do :** Check the ITDR dashboard within the **Defender XDR** portal for any signs of identity-based threats or anomalies in user behavior.
 - **Why it matters :** The ITDR dashboard offers a holistic view of identity risks. Keeping an eye on this dashboard helps you identify and respond to potential breaches early.
- **Perform proactive threat hunting :** Being proactive with advanced hunting allows you to seek out threats before they trigger alerts. This method keeps you one step ahead of attackers and helps identify vulnerabilities that need attention:
 - **What to do :** Use advanced hunting within the **Defender XDR** portal to search for potential indicators of compromise or other suspicious activity. Start using a model as described in [Chapter 9](#).
 - **Why it matters :** Hunting for threats that haven't yet triggered an alert allows you to stay ahead of attackers who might be probing your defenses. You can also see gaps in your infrastructure that need attention, including everything from adding a new Conditional Access policy to seeing network traffic that should be blocked between your tiering model in your on-premises infrastructure.
- **Check sensor health :** Ensuring that MDI sensors are functioning properly is key to effective threat detection. A quick health check will reveal any issues that could compromise your security visibility:
 - **What to do :** Verify the status of your MDI sensors through the Health Center. Address any sensor health issues immediately to ensure continued data collection.
 - **Why it matters :** Sensors are the backbone of your MDI system. If they aren't functioning correctly, critical security events may go unnoticed, leaving gaps in your defenses.

After handling the essential daily checks to ensure your MDI system is operating efficiently, it's time to shift focus toward slightly more strategic tasks that help fortify your security posture over the course of a week. Weekly tasks allow for deeper reviews, giving you the opportunity to assess evolving threats, monitor detection performance, and proactively adjust your settings to stay ahead of risks. Let's dive into the key actions that should form part of your weekly routine.

Weekly tasks

Your *weekly routine* should emphasize improving your security posture and adapting to the evolving threat landscape. This is the time to review performance, assess strategic recommendations, and take proactive steps to safeguard your MDI environment from emerging risks. Coming tasks help refine your defenses, stay ahead of new threats, and ensure your security measures evolve alongside the changing landscape:

- **Evaluate secure score recommendations** : Regular reviews of your Secure Score provide actionable insights for enhancing your security. Implementing these recommendations ensures your defenses remain aligned with the latest best practices:
 - **What to do** : Assess your Secure Score recommendations within the **Defender XDR** portal and implement the recommended actions that will improve your organization's security posture. You can easily filter the recommendations by product, such as Microsoft Defender for Identity.
 - **Why it matters** : Following the recommendations from Secure Score helps you take proactive steps to close security gaps and protect against identity-related vulnerabilities, ensuring your organization's defenses are up to par with best practices.
- **Monitor and respond to new threats** : New threats are constantly emerging, and it's important to stay ahead of them. Keeping up with the latest threat intelligence and comparing it to your own environment allows you to adapt quickly to new risks:
 - **What to do** : Stay updated on emerging cybersecurity threats by reviewing threat intelligence reports and assessing how these developments might affect your environment. Compare this information with your MDI data to determine if there are any signs of potential compromise.
 - **Why it matters** : The threat landscape is always evolving, with new attack vectors and vulnerabilities appearing. By regularly monitoring new threats, you can adapt your security measures and detection rules to better defend against these risks.
- **Perform proactive threat hunting** : By actively searching for threats before they trigger alerts, you can identify and mitigate potential vulnerabilities, ensuring that attackers don't have the opportunity to exploit weaknesses:
 - **What to do** : Use advanced hunting tools within Defender XDR to search for potential indicators of compromise or other suspicious activity.
 - **Why it matters** : Hunting for threats that haven't yet triggered an alert allows you to stay ahead of attackers who might be probing your defenses.

Now that the weekly checks have strengthened your system's foundation, it's time to focus on more comprehensive monthly maintenance. These tasks help ensure that your MDI deployment remains optimized over the long term, allowing you to refine alerts and stay updated with the latest changes in Microsoft's Defender ecosystem. Let's explore what needs attention in your monthly routine.

Monthly tasks

Your *monthly routine* is an opportunity to *fine-tune* existing configurations and stay updated on new developments within the Microsoft Defender XDR suite. These tasks ensure your MDI system is optimized and keeps up with the evolving cybersecurity landscape. By focusing on both immediate

performance and the latest changes, you'll help secure your environment and build resilience against future threats:

- **Review and adjust tuned alerts** : Start by reexamining the alerts you've refined in recent weeks:
 - **What to do** : Revisit the detection alerts that you've adjusted in previous weeks. Evaluate whether the current settings are still effective in minimizing false positives while catching legitimate threats. If your security needs have changed or new types of false positives have emerged, adjust the tuning accordingly.
 - **Why it matters** : Threats evolve, and your detection system should evolve with them. Regularly reviewing and fine-tuning your alerts ensures that they remain accurate and focused, preventing unnecessary noise while maximizing your ability to spot real threats.
- **Stay informed about new changes in Microsoft Defender XDR and Defender for Identity** : After refining your detection system, it's essential to stay updated with the latest changes to ensure your defenses are up-to-date:
 - **What to do** : Track the latest updates, features, or changes released by Microsoft in the Defender XDR and MDI products. Make sure your team understands how to implement new features and take advantage of improvements in threat detection, reporting, and overall functionality.
 - **Why it matters** : Microsoft frequently releases updates and new capabilities in Defender XDR and MDI to enhance threat detection and response. By staying informed about these changes, you can ensure that your environment is always operating with the latest security innovations and optimizations.

As your monthly maintenance ensures that your MDI system remains optimized and up to date, it's also important to occasionally step back and perform more in-depth reviews of your infrastructure. This is where quarterly or ad-hoc tasks come into play.

Quarterly/ad-hoc tasks

While daily and weekly tasks keep your MDI environment running smoothly in the short term, it's essential to periodically dive deeper into broader system reviews. Quarterly or ad-hoc tasks focus on maintaining long-term stability, adjusting configurations as your infrastructure evolves, and addressing any emerging risks. These tasks ensure your MDI deployment is continuously aligned with your organization's security objectives. Quarterly reviews play a critical role in fine-tuning your system and ensuring your infrastructure keeps pace with ongoing changes:

- **Regularly monitor Microsoft service health** : Checking the service health status of key Microsoft services ensures that your environment stays functional and responsive:
 - **What to do** : Periodically review the Microsoft service health dashboard to stay updated on the operational status of critical services such as MDI, Entra ID, and Microsoft Defender XDR. Monitor for any outages or incidents that could affect your environment's security monitoring capabilities.
 - **Why it matters** : External service disruptions can hinder the performance and reliability of your MDI deployment. Keeping a close eye on service health helps you proactively address potential impacts before they affect your security operations.

- **Review server setup and ensure sensor deployment** : As your network infrastructure evolves, reviewing server setups ensures comprehensive coverage and optimal detection capabilities:
 - **What to do** : Review your server infrastructure to verify that all critical domain controllers and other infrastructure components are equipped with MDI sensors. Ensure that newly added servers are correctly configured and reporting to MDI.
 - **Why it matters** : As your infrastructure evolves, certain servers may be overlooked, leaving gaps in your monitoring coverage. By regularly reassessing your server setup, you maintain comprehensive detection across your entire network.
- **Validate domain configuration via PowerShell Scripts** : Automated validation of domain settings ensures the system is running with proper configurations, preventing misconfigurations that may arise over time:
 - **What to do** : Run scheduled PowerShell scripts to check domain controller configurations. Ensure that settings such as gMSA permissions, port mirroring, and network configurations are correctly applied to maintain smooth communication between MDI sensors and the backend.
 - **Why it matters** : Misconfigurations or network issues can disrupt the data flow to your MDI system. Regularly validating domain controller configurations with PowerShell helps catch issues early and keeps your environment functioning optimally.

I cannot stress enough the importance of watching the Secure Score recommendations because the valuable information that you get from the insights and telemetry from all of the Defender products helps you close the gap. Try to prioritize this work. It can perhaps feel daunting when you see the list of items that need to be addressed, but I can assure you that it will help you in the long run. A tip is to look for items that can be implemented fast.

Let's summarize the final chapter of the book!

Summary

In this chapter, we explored the key aspects of troubleshooting and optimizing your Microsoft Defender for Identity deployment. This included various items, from seeing the signs of trouble, using the built-in logs, and checking communication, to daily maintenance tasks that help keep your system running smoothly, as well as weekly, monthly, and quarterly checks that ensure long-term stability. Each step plays a vital role in maintaining a secure and efficient environment. The importance of regularly reviewing and fine-tuning your settings, proactively hunting for threats, and addressing any potential issues early, is to ensure that your MDI system continues to protect your organization from identity-based threats.

Future reading

- Microsoft's list of known issues in MDI: <https://learn.microsoft.com/en-us/defender-for-identity/troubleshooting-known-issues>

Index

As this ebook edition doesn't have fixed pagination, the page numbers below are hyperlinked for reference only, based on the printed edition of this book.

A

AbuseIPDB [309](#)

access control entry (ACE) [102](#)

access token

 obtaining [150](#)

 obtaining, with Postman [150](#) , [151](#)

 obtaining, with PowerShell [151](#) , [154](#) - [156](#)

Active Directory (AD) [3](#) , [16](#) , [33](#) , [199](#) , [239](#) , [268](#) , [294](#)

Active Directory Certificate Services (AD CS) [11](#) , [17](#) , [62](#) , [95](#) , [109](#)

 configuring, for MDI sensor installation [113](#)

 used, for integrating MDI [109](#) - [111](#)

 working [112](#)

Active Directory Federation Services (AD FS) [11](#) , [17](#) , [62](#) , [95](#)

 advanced auditing [100](#)

 authentication, working [98](#) , [99](#)

 configuring, for MDI sensor installation [99](#)

 database permissions [106](#)

 SACL configuration [101](#)

 used, for integrating MDI [97](#)

 verbose settings [100](#)

Active Directory Service Accounts (AD SAs) [31](#) , [33](#) , [49](#)

 group managed service accounts (gMSAs) [32](#) , [50](#) , [51](#)

 in Defender for Identity [33](#) , [34](#)

KDS root key [50](#)

KDS root key for gMSAs, need for [49](#)

verifying [56](#)

Active Directory Users and Computers [103 - 105](#), [274](#)

AD attack paths [199](#)

- credential theft [199](#)
- lateral movement [199](#)
- persistence [199](#)
- privilege escalation [199](#)

AD CS advanced auditing [114](#)

- CA auditing, configuring with GUI [116](#), [117](#)
- CA auditing, implementing via PowerShell [115](#), [116](#)
- enabling, through Group Policy [114](#), [115](#)

AD CS integration

- validating [117 - 121](#)

AD FS integration

- validating [108](#), [109](#)

AdminSDHolder [271](#)

advanced attacks with KQL

- detecting [204](#)
- Kerberoasting [211](#)
- prerequisites [205](#)

advanced attacks with KQL, prerequisites

- Defender XDR advanced hunting [209](#)
- Microsoft Defender, deploying [208](#)
- Mimikatz, installing [206](#)
- PtH attack [209](#)

workstations, joining to domain [206](#)

advanced KQL techniques for deep threat detection [199](#)

AD attack paths [199](#)

crafting [201](#)

domain controllers not as file servers, ensuring [203](#)

enumeration attacks, detecting [203](#)

Kerberos sign-ins, identifying [201](#)

kill chain [199](#) , [200](#)

legacy service accounts, monitoring [201](#)

multiple service accounts, monitoring [202](#)

NTLM, identifying [201](#)

service accounts, identifying [202](#)

advanced persistent threats (APTs) [9](#) , [242](#)

advanced PowerShell scripts for MDI management [73](#) - [75](#)

health issues API [75](#) - [79](#)

advanced threat [243](#)

defining [243](#) - [245](#)

incident detection and validation [248](#)

response strategy and execution [255](#)

advanced threat detection [248](#)

advanced threat tactics

pre-incident preparation [247](#) , [248](#)

advanced threat tactics, examples

code injection [247](#)

Distributed Denial-of-Service (DDoS) attacks [246](#)

identity-based attacks [246](#)

insider threat [246](#)

Internet of Things (IoT) attacks [247](#)

malware [246](#)

phishing campaigns [246](#)
ransomware [246](#)
supply chain attacks [247](#)

alert [237](#)
versus incident [237](#)

Alert API endpoint
examples [163](#), [164](#)
working with [163](#)

alert tuning [225](#)
alert thresholds in MDI [228](#), [229](#)
in Defender XDR [226](#), [227](#)
rules, re-evaluating [227](#), [228](#)

API calls
Alert API endpoint, working with [163](#)
healthIssue API endpoint, working with [160](#)
incidents, investigating with Incident API endpoint [165](#)
making [158](#), [159](#)
application object [139](#)
app registration [139](#)
with Azure CLI [146](#)
with Azure PowerShell [147](#), [148](#)
artificial intelligence [249](#)

Attack Disruption feature [322](#)

attackers characteristics
high impact [246](#)
multi-vector approach [246](#)
persistence [246](#)
sophistication [245](#)

targeted nature [246](#)

auditing object [102](#)

- key attributes [102](#)

Authentication, Authorization, and Accounting (AAA) [132](#)

Automatic Attack Disruption [270](#), [280](#)

automation build [258](#)

- continuous deployment (CD) [259](#)
- continuous integration (CI) [258](#)
- KQL detections [259](#)

Azure Active Directory [270](#)

Azure Advanced Threat Protection (ATP) [17](#)

Azure Arc security [82](#)

Azure Automation [257](#)

Azure CLI

- using [146](#)

Azure Data Explorer (ADX) [173](#), [174](#)

- foreign key [174](#)
- primary key [174](#)
- queries [175](#), [176](#)

Azure Durable Function [307](#) - [309](#)

Azure Event Grid [307](#)

Azure ExpressRoute [21](#)

Azure Functions [258](#)

Azure Key Vault [139](#)

Azure Logic Apps [307](#)

Azure Monitor [247](#)

- used, for monitoring MDI service [80](#) - [82](#)

Azure Monitor Agent (AMA) [61](#)

Azure PowerShell

using [147](#), [148](#)

Azure Resource Manager (ARM) [143](#)

B

Benign True Positive (B-TP) [224](#)

example [224](#)

bicepconfig.json

reference link [143](#)

BloodHound [301](#)

C

centralized IRT [261](#)

advantages [261](#)

challenges [261](#)

structure and composition [261](#)

Certificate Authority (CA) [109](#)

Certificate-Based Authentication (CBA) [97](#)

Certificate Revocation List (CRL) [112](#)

Certificate Servers

Microsoft Defender for Identity (MDI), need for [113](#)

Certificate Signing Request (CSR) [109](#)

certificate template [109](#)

classification outcomes [223](#)

cloud access security broker (CASB) [131](#)

Cloud Application Administrator [141](#)

Cloud Solution Providers (CSPs) [4](#)

Common Event Format (CEF) [88](#)

computer accounts [32](#)

confusion matrix [223 - 225](#)

Benign True Positive (B-TP) [224](#)

false negative (FN) [224](#)

false positive (TP) [224](#)

informational alerts [224](#)

true negative (TN) [224](#)

true positive (TP) [224](#)

continuous deployment (CD) [259](#)

key components [259](#)

continuous integration and continuous deployment (CI/CD) [258](#)

continuous integration (CI) [258](#)

coordinated IRT [262](#)

advantages [262](#)

challenges [262](#)

structure and composition [262](#)

credential phishing attacks [270](#)

credentials not retrieved [323](#)

credential theft [281](#)

credential theft attack

simulating, steps [282](#)

simulating, with Mimikatz [282 - 285](#)

custom alert rules

used, for monitoring MDI configuration [82 - 87](#)

custom detection rule [285](#)

creating [285 , 286](#)

custom gMSA [270](#)

custom integrations and automations

building [166](#)

integration opportunities, identifying [166](#)
use case types [167](#)
custom malware [243](#)
cutting-edge technology [243](#)
Cyber Kill Chain [6](#)
cyberattack stages [6](#)

D

database administrator (DBA) [106](#)
data collection [25](#), [26](#)
Data Collection Endpoint (DCE) [82](#), [167](#)
data collection rule (DCR)
Data Collection Rule (DCR) [81](#), [167](#)
data flow
securing [131](#)
DCShadow attack [204](#), [213](#)
detecting [216](#)
executing [215](#)
mitigation and detection strategies [214](#)
vector [214](#)
working [213](#)
Defender for Endpoint [248](#)
Defender for Identity license [17](#)
Defender for Identity PowerShell module [327](#)
Defender for Office 365 [248](#)
Defender XDR [4](#), [220](#)
Defender XDR Advanced Hunting [331](#)
Defender XDR Attack Disruption [277](#)
Defender XDR portal [56](#), [57](#), [269](#), [301](#)

group mapping in Entra ID [57 - 59](#)
role definition [57](#)

Defender XDR Unified RBAC [276](#)

deployment logs [321](#)

detection [245](#)

Development Operations (DevOps) [258](#)

Directory Services Restore Mode (DSRM) [314](#)

disaster recovery [304](#)
for identity systems [312 - 315](#)

distributed IRT [261](#)
advantages [262](#)
challenges [262](#)
structure and composition [261](#)

Document Object Model (DOM) [326](#)

domain controller (DC) [213 , 268](#)

Durable Function [307](#)

E

effective queries [193](#)

enable logging [247](#)

endpoint detection and response (EDR) [306](#)

Enterprise Access Model

reference link [11](#)

enterprise application [139 , 140](#)

Enterprise CA [109](#)

Enterprise Mobility + Security E5 (EMS E5/A5) [17](#)

Entra Connect [62 , 95](#)

advanced auditing [123 , 124](#)

configuring, for MDI sensor [123](#)

integration, validating [124](#), [126](#)
used, for integrating Microsoft Defender for Identity (MDI) [121](#)
working [122](#)

Entra ID [270](#)

event ID 4662 [279](#)

event ID 4725 [281](#)

event logs [321](#)

Extended Detection and Response (XDR) [4](#)

Extended/Enhanced Key Usage (EKU) [109](#)

Extended Security Updates (ESUs) [19](#), [66](#)

extension [144](#)

F

false negative (FN) [224](#)
example [224](#)

false positive (TP) [224](#)
example [224](#)

Federated Identity [97](#)

full outer join [187](#)

fully qualified domain name (FQDN) [194](#)

G

general availability (GA) [75](#)

Globally Unique Identifier (GUID) [102](#)

golden gMSA attacks [279](#)

granular access control [131](#)

Graph API [307](#)
need for [139](#)

group managed service accounts (gMSAs) [31](#), [32](#), [50](#), [51](#), [201](#), [270](#), [323](#)

need for 33
scalability and reliability 32
security 32
simplified management 32
setting, in Defender XDR portal 55
Group Policy Management Console (GPMC) 114
Group Policy Object (GPO) 52, 114, 254, 301
Group Policy Tampering 254, 255

H

healthIssue API endpoint
working with 160
healthIssue API endpoint, methods
get method 161, 162
list method 160, 161
update method 162, 163
health issues
sending, via syslog to Microsoft Sentinel 88 - 91
Health Issues page 319
hunting tables 124
usage 198
hybrid IRT 263
advantages 263
challenges 264
structure and composition 263
hypothesis-driven threat hunting 294, 295
ExposureGraphEdges 295
ExposureGraphNodes 296
IdentityDirectoryEvents 295

IdentityInfo [295](#)
IdentityLogonEvents [295](#)
IdentityQueryEvents [295](#)
identity threats [296](#)

I

IAM solution [4](#)
identity and access management (IAM) [4](#), [132](#)
identity-based incidents with SOAR
activity function [311](#)
information gathering, with Azure Durable Function [309](#), [310](#)
orchestrator function [310](#), [311](#)
response, automating [306](#) - [309](#)
IdentityDirectoryEvents [194](#) - [196](#)
key columns [194](#)
identity-hardening practices [300](#)
approach [300](#)
continuous improvement [301](#)
vulnerabilities [300](#)
identity-hardening strategies
best practices [303](#), [304](#)
implementing [301](#) - [303](#)
IdentityInfo [196](#), [197](#)
IdentityLogonEvents [193](#), [194](#)
identity plane [294](#)
IdentityQueryEvents [197](#), [198](#)
Identity Threat Detection and Response (ITDR) [3](#), [4](#), [289](#), [335](#)
MDI, implementing [5](#)
identity threats [296](#)

service creation events for persistent threats, monitoring [298](#) , [299](#)

unauthorized group membership change, detecting [296](#) - [298](#)

incident [237](#)

versus alert [237](#)

Incident API endpoint

used, for investigating incidents [165](#) , [166](#)

incident detection and validation [248](#)

Incident Response Plan (IRP) [264](#) , [304](#)

defining [305](#) , [306](#)

Incident Response Team (IRT) [260](#)

building [260](#)

centralized IRT [261](#)

coordinated IRT [262](#)

detection and analysis [264](#)

distributed IRT [261](#)

hybrid IRT [263](#)

preparation [264](#)

incidents

investigating, with Incident API endpoint [165](#)

indicators of compromise (IOCs) [226](#)

informational alerts [224](#)

example [224](#)

Infrastructure as Code (IaC) [142](#)

initial triage and categorization [236](#)

alert or incident, receiving [237](#) - [239](#)

alerts by severity and potential impact, categorizing [240](#)

alerts, prioritizing on criticality and context [240](#)

automated triage processes, implementing [241](#)

inner join [185](#)

inner unique join [187](#) , [188](#)

integration opportunities

example [166](#) , [167](#)

identifying [166](#)

intrusion detection system (IDS) [264](#)

containment, eradication, and recovery [264](#)

post-incident activity [265](#)

Invoke-RestMethod [151](#)

features [152](#)

usage [152](#)

Invoke-WebRequest [151](#)

features [152](#)

usage [153](#)

ITDR posture [289](#)

elevating, with MDI [300](#)

identity-hardening practices [300](#)

identity-hardening strategies, implementing [301](#) - [303](#)

J

join types [184](#)

full outer join [187](#)

inner join [185](#)

inner unique join [187](#) , [188](#)

left anti join [188](#)

left outer join [186](#)

left semi join [189](#)

right anti join [188](#)

right outer join [186](#)

right semi join [189](#)

K

Kerberoasting [204](#), [211](#)

attack vector [212](#)

detecting [213](#)

executing [212](#), [213](#)

mitigation and detection strategies [212](#)

working [211](#)

Kerberos ticket consideration [128](#)

Key Distribution Service (KDS) [49](#)

KQL detections [259](#)

need for [259](#)

KQL query [331](#)

Kusto

origins [173](#)

Kusto Detective Agency (KDA) [176](#)

features [176](#)

Kusto Query Language (KQL) [171](#), [178](#) - [181](#), [259](#), [289](#)

broader applications and ecosystem [176](#)

columns, renaming [182](#)

data, combining [192](#), [193](#)

data, filtering [183](#)

data, grouping [192](#), [193](#)

data, sorting [183](#)

development [173](#)

ecosystem [173](#)

evolution [173](#)

history [173](#)

multiple tables, combining [183](#)

new columns, defining [182](#)

used, for exploring data [181](#)

L

lateral movement paths (LMPs) [234 - 236](#)

LDAP [300](#)

left anti join [188](#)

left outer join [186](#)

left semi join [189](#)

left table [183](#)

LocalSystem [271](#)

LocalSystem account [268](#)

Log Analytics [247](#)

Logic App [307](#)

Logic Apps [257](#)

lower threshold [334](#)

M

machine learning [249](#)

malfunctioning MDI sensor

 network connectivity problems, troubleshooting [326](#)

 removing, manually [324 - 326](#)

Managed Detection and Response (MDR) [271](#)

Managed Security Service Provider (MSSP) [270](#)

Managed Service Accounts (MSAs) [212](#)

Managed Service Providers (MSPs) [4](#)

MDI action accounts

 automated threat response [280 , 281](#)

configuration, best practices [271 - 278](#)

configuring [268](#)

credential theft attack, simulating [282 - 285](#)

credential theft, detecting and responding [281](#)

custom detection rule, creating [285](#), [286](#)

in multi-forest environments [277](#)

lateral movement, detecting and responding [281](#)

need for [268 - 270](#)

operational efficiency [285](#)

real-world situations [280](#)

securing [268](#)

security measures [278](#), [279](#)

selecting, between system or custom gMSAs [270](#), [271](#)

use cases [280](#)

MDI alert settings

adjusting, for better accuracy [333](#), [334](#)

detection rules, customizing [332](#)

filtering techniques, applying [332](#)

MDI alert system [220 - 223](#)

alert tuning [225](#)

confusion matrix [223 - 225](#)

reference link [221](#)

MDI API [138](#)

MDI checklist

Active Directory service accounts [31](#)

AD CS, prerequisites [30](#), [31](#)

AD FS, prerequisites [30](#), [31](#)

data collection [25](#), [26](#)

licenses [17](#), [18](#)

networking [21](#)

operating system requirements [18](#) - [20](#)

permissions [18](#)

planning [17](#)

PowerShell [23](#), [24](#)

pre-installation [17](#)

sensor requirements [20](#)

sizing [27](#)

user profiling [26](#), [27](#)

MDI configuration

monitoring, with custom alert rules [82](#) - [87](#)

MDI data [177](#)

hunting tables, usage [198](#)

key data types [177](#)

querying [172](#)

table concepts [177](#), [178](#)

MDI deployment [34](#), [35](#)

access key [37](#) - [39](#)

daily tasks [335](#), [336](#)

installation package [37](#), [39](#)

lab environment [35](#) - [37](#)

MDI sensor, configuring [34](#)

MDI sensor, installing [34](#)

monthly routine [337](#), [338](#)

operational guide [334](#)

quarterly/ad-hoc tasks [338](#), [339](#)

security alert misfires, resolving [332](#)

sensor, installing with PowerShell [43 - 45](#)

sensor, installing with UI [40 - 42](#)

validation [35](#)

weekly routine [336 , 337](#)

MDI healthIssues

integrating, with monitoring tools [167 - 170](#)

MDI Health Issues page [332](#)

MDI hunting tables [193](#)

IdentityDirectoryEvents [194 - 196](#)

IdentityInfo [196 , 197](#)

IdentityLogonEvents [193 , 194](#)

IdentityQueryEvents [197 , 198](#)

MDI incident response scenarios [250](#)

Group Policy Tampering [254 , 255](#)

Suspected Additions to Sensitive Groups [251 , 252](#)

Suspected Suspicious Kerberos Ticket Request [250](#)

Suspicious Additions to Sensitive Groups [253](#)

MDI issues

configuration settings, checking [322](#)

diagnosing [318](#)

errors and fixes, troubleshooting [322 - 324](#)

signs of trouble, spotting [318 , 319](#)

tools and logs, used for searching [319](#)

MDI issues, tools and logs

event logs [321 , 322](#)

Microsoft Defender for Identity Health Center [319 - 321](#)

MDI licensing requirements

reference link [18](#)

MDI PowerShell module [23](#), [62](#)

file, overview [65](#)

functions [66](#) - [73](#)

importing, on sensor servers without internet access [63](#) - [65](#)

installing [63](#)

installing, on sensor servers with internet access [63](#)

reference link [23](#)

using [327](#), [328](#)

MDI proxy configuration

navigating [45](#), [46](#)

MDI sensor [113](#)

MDI sensor connectivity

MDI PowerShell module, using [327](#), [328](#)

PowerShell, using [326](#), [327](#)

testing [326](#)

MDI sensor in multi-forest

example design [128](#)

Kerberos ticket consideration [128](#)

MDI sensor in multi-forest, design example

configuration [130](#)

deployment strategy [129](#), [130](#)

scenario overview [129](#)

security measures [130](#)

MDI sensor in multi-forest prerequisites [127](#)

universal group consideration [128](#)

MDI sensor installation

used, for configuring AD CS [113](#)

MDI sensors

NNR issues, troubleshooting [330 - 332](#)

proxy configuration issues [328 , 329](#)

SSL inspection causing issues [330](#)

MDI service

- monitoring, via Azure Monitor [80 - 82](#)
- reference link [22 , 23](#)

MDI Sizing Tool [27](#)

- reference link [27](#)

methodical approach

- developing, to alert investigation [220](#)

Microsoft 365 [16](#)

- Microsoft 365 E3 + E5 Security [16](#)
- Microsoft 365 E5 [16](#)
- Microsoft 365 E5/A5/G5 Security [17](#)
- Microsoft 365 E5 (Microsoft E5/A5/G5) [17](#)
- Microsoft 365 suite [293](#)
- Microsoft Copilot for Security [238](#)
- Microsoft Defender [220](#)
- Microsoft Defender for Endpoint (MDE) [21 , 171](#)
- Microsoft Defender for Identity Health Center [319](#)
- Microsoft Defender for Identity (MDI) [3 , 15 - 17 , 61 , 95 , 219 , 267 , 289 , 317](#)
- benefits [13 , 14](#)
- expanding, across multiple Active Directory forests [126](#)
- high-level architecture [12 , 13](#)
- importance, on Certificate Servers [112 , 113](#)
- integrating, with AD CS [109 - 111](#)
- integrating, with AD FS [97](#)
- integrating, with Entra Connect [121](#)

key features [13](#) , [14](#)

strategic position, in cybersecurity ecosystem [11](#)

Microsoft Defender XDR portal [237](#)

Microsoft Entra ID [4](#)

Microsoft Entra-joined devices [52](#)

Microsoft Entra Private Access [131](#)

Microsoft Graph [138](#)

Microsoft Graph API [137 - 141](#)

- access token, obtaining [150](#)
- API calls, making [158](#) , [159](#)
- app registration, creating with Bicep [142 - 145](#)
- app registration, creating with Microsoft Entra portal [141](#) , [142](#)
- enterprise application, creating [148](#) , [149](#)
- security and compliance [156](#)

Microsoft identity platform

- API keys and secrets [157](#) , [158](#)
- reference link [156](#)

Microsoft Incident Response guides [293](#)

Microsoft Incident Response Ninja Hub [289](#)

Microsoft Incident Response teams [293](#)

Microsoft Intune admin center

- reference link [53](#)

Microsoft RRAS

- configuring [133](#)
- installing [133](#)
- monitoring [134](#) , [135](#)
- verifying [134](#) , [135](#)

Microsoft Security Response Center (MSRC) [19](#)

Microsoft Sentinel [4](#), [61](#), [239](#)

health issues, sending via syslog [88](#) - [91](#)

security alerts, sending via syslog [88](#) - [91](#)

Microsoft tenant [16](#)

Microsoft Threat Intelligence Center (MSTIC) [249](#)

Mimikatz [8](#)

downloading [207](#)

files, extracting [208](#)

MDE antivirus temporarily, disabling [207](#)

running [208](#)

used, for simulating credential theft attack [282](#) - [285](#)

MITRE ATT&CK framework [7](#)

core components [7](#)

Enterprise Matrix [7](#), [8](#)

real-world scenarios [8](#)

modern identity threats and strategic defense frameworks [5](#)

Cyber Kill Chain [6](#), [7](#)

MITRE ATT&CK framework [7](#) - [9](#)

techniques [5](#), [6](#)

Unified Kill Chain [9](#) - [11](#)

monitoring and logging [279](#)

multi-factor authentication (MFA) [4](#), [210](#), [247](#), [270](#), [293](#)

multiple Active Directory forests

concept [126](#), [127](#)

MDI expanding [126](#)

trusts types [127](#)

multiple tables

join types [184](#), [185](#)

left and right tables [183](#) , [184](#)

scenario 2 example [191](#)

scenario example [189](#) , [190](#)

N

National Institute of Standards and Technology (NIST) [260](#) , [305](#)

networking [21](#)

Azure ExpressRoute [21](#)

firewall [21](#)

Forward proxy [21](#)

ports [21](#) , [22](#)

reference link [22](#) , [23](#)

Network Name Resolution (NNR)

issues, troubleshooting in MDI [330](#) - [332](#)

Network Time Protocol (NTP) [247](#)

new gMSAs [279](#)

new root KDS key [279](#)

New Technology LAN Manager (NTLM) [67](#) , [201](#) , [300](#)

Npcap [39](#) , [324](#)

Npcap 1.0 OEM [39](#)

O

object identifiers (OIDs) [109](#)

offline address book (OAB) [52](#)

offline Root CA [113](#)

Okta [4](#)

Online Certificate Status Protocol (OCSP) [112](#)

optimization [317](#)

Organizational Unit (OU) [124](#) , [270](#)

outbound [21](#)
outsourced IRT [263](#)
 advantages [263](#)
 challenges [263](#)
 structure and composition [263](#)

P

packet capture (PCAP) [39](#)
pass-the-hash (PtH) [204](#), [300](#)
Pass-the-Ticket [300](#)
Pass-Through Authentication (PTA) [97](#), [122](#)
Password Hash Synchronization (PHS) [97](#), [122](#)
People, Process, Technology (PPT) [244](#)
PingCastle [302](#)
Platform Identity Management [80](#)
Pointer (PTR) records [331](#)
ports [21](#), [22](#)
post-installation activities [48](#)
 Active Directory Service Accounts (AD SAs) [48](#), [49](#)
 AD SAs, verifying [56](#)
 Defender XDR portal [56](#), [57](#)
 gMSA, setting in Defender XDR portal [55](#)
 RBAC [49](#)
 SAM-R [48](#)
 SAM-R, configuring [52](#) - [55](#)
Postman
 used, for obtaining access token [150](#), [151](#)
Power Automate [257](#)
PowerShell [16](#), [23](#), [24](#), [101](#), [244](#), [274](#), [307](#)

used, for obtaining access token [151](#) , [154 - 156](#)
using [326](#) , [327](#)

PowerShell script code
reference link [308](#)

principle of least privilege [247](#)

privileged access deployment
reference link [313](#)

privileged access workstations (PAWs) [214](#) , [303](#)

proactive identity security posture assessments
reference link [303](#)

proactive threat-hunting [290](#)

proactive threat-hunting strategies
accurate detection [292](#)
behavioral analytics, using [294](#)
designing, with MDI [290](#)
hypothesis-driven threat hunting [294](#) , [295](#)
logging [292](#)
methodology [290](#) , [291](#)
security use case development [293](#) , [294](#)

proper configuration of security tools [247](#)

provider [144](#)

PtH attack [204](#) , [209](#)
detecting [211](#)
executing [210](#) , [211](#)
mitigation and detection strategies [210](#)
vector [210](#)
working [209](#)

Public Key Infrastructure (PKI) [109](#)

Purple Knight [302](#)

Q

quarterly/ad-hoc tasks [338](#), [339](#)

R

reduced latency [131](#)

regular training [247](#)

relational database management system (RDBMS) [175](#)

remediation process [292](#)

remote activities

securing [131](#)

Remote Authentication Dial-In User Service (RADIUS) [131](#), [132](#)

implementation [132](#), [133](#)

integrating [134](#)

logs, using to configure MDI [134](#)

Remote Desktop Protocol (RDP) [254](#)

response strategy and execution [255](#)

automation build [258](#)

key components [256](#)

SOAR tools in Microsoft Cloud [257](#)

reverse DNS lookup (UDP Port 53) [331](#)

RFC 3164 [90](#)

versus RFC 5424 [90](#)

RFC 5424

versus RFC 3164 [90](#)

right anti join [188](#)

right outer join [186](#)

right semi join [189](#)

right table [184](#)

robust detection mechanisms [247](#)

robust password policy [6](#)

Role-Based Access Control (RBAC) [18](#), [80](#), [270](#)

Root Cause Analysis (RCA) [241 - 243](#)

Routing and Remote Access Server (RRAS) [96](#), [132](#)

limitations [132](#)

S

SACL for advanced auditing settings

configuring [103](#)

Safe Links [270](#)

SAM-R

configuring [52 - 55](#)

SAM-R call [34](#)

secure admin workstations (SAWs) [214](#)

secure gMSA [279](#)

Secure Score page [301](#)

BloodHound [301](#)

PingCastle [302](#)

Purple Knight [302](#)

secure web gateway (SWG) [131](#)

Security Account Manager-Remote (SAM-R) [48](#)

Security Administrator [18](#)

security alert misfires

resolving [332](#)

security alerts

sending, via syslog to Microsoft Sentinel [88 - 91](#)

Security Filtering [124](#)

security information and event management (SIEM) [157](#), [245](#)

Security Operations (SecOps) [258](#)

Security Orchestration, Automation, and Response (SOAR) [241](#), [257](#), [306](#)

security professionals [248](#)

Security Service Edge (SSE) [131](#)

security tools [248](#)

security use case development [292](#) - [294](#)

streamlined approach [293](#)

sensitive entities [252](#)

service accounts, types

computer accounts [32](#)

group managed service accounts (gMSAs) [31](#)

standalone managed service accounts (sMSA) [31](#)

user accounts [32](#)

Service Level Agreement (SLA) [18](#)

service principal name (SPN) [31](#), [34](#), [211](#), [250](#)

simplified security infrastructure [131](#)

single-label domains (SLDs) [56](#)

Single Sign-On (SSO) [97](#), [122](#)

sizing [27](#)

estimation, for domain controllers [27](#) - [29](#)

SOAR tools in Microsoft Cloud [257](#)

Azure Automation [257](#)

Azure Functions [258](#)

Logic Apps [257](#)

Power Automate [257](#)

Source of Authority (SoA) [11](#)

standalone managed service accounts (sMSA) [31](#)

Standard Operating Procedures (SOPs) [245](#)
strict access controls [279](#)
Structured Query Language (SQL) [175](#)
Subject Alternative Name (SAN) [110](#)
Suspected Suspicious Kerberos Ticket Request [250](#), [251](#)
Suspicious Additions to Sensitive Groups [251](#) - [253](#)
synchronization [122](#)
syslog
 used, for sending health issues to Microsoft Sentinel [88](#) - [91](#)
 used, for sending security alerts to Microsoft Sentinel [88](#) - [91](#)
system access control list (SACL) [67](#), [100](#)
system account [270](#)

T

tactics, techniques, and procedures (TTPs) [248](#)
Test-MdiReadiness.ps1
 reference link [24](#)
test mode [26](#)
threat-hunting strategies [289](#)
threat intelligence [249](#)
threat landscape [4](#)
Ticket Granting Service (TGS) [211](#), [250](#)
Ticket Granting Tickets (TGTs) [6](#), [198](#), [250](#)
TinyProxy [45](#)
 benefits [46](#)
 configuring [47](#), [48](#)
 installing [46](#), [47](#)
troubleshooting [317](#)
true negative (TN) [224](#)

example [224](#)
true positive (TP) [224](#)
example [224](#)
trusts types
in multiple Active Directory forests [127](#)
non-transitive trust [127](#)
one-way trust [127](#)
transitive trust [127](#)
two-way trust [127](#)

U

Ubuntu 22.04 [16](#)
Unified Kill Chain (UKC) [9](#) - [11](#)
phases [9](#), [10](#)
reference link [11](#)
universal group consideration [128](#)
global catalog load [128](#)
security [128](#)
update logs [321](#)
UseBasicParsing parameter [326](#)
use case engineering [292](#)
use case modeling [292](#)
user accounts [32](#)
User Entity [229](#), [230](#)
Incidents and Alerts tab [231](#), [232](#)
Observed in Organization tab [232](#)
Overview tab [230](#), [231](#)
Policies tab [233](#)
Timeline tab [233](#)

User Entity Behavior Analytics (UEBA) [13](#)

User Principal Name (UPN) [110](#), [194](#)

user profiling [26](#), [27](#)

V

validation process [249](#)

virtual machines (VMs) [80](#)

virtual or physical server environment [16](#)

Virtual Private Networks (VPNs) [95](#)

VPN integration [131](#)

W

Web Application Proxy (WAP) [30](#), [97](#)

Windows events

reference link [26](#)

Windows Internal Database (WID) [106](#)

Windows NT [67](#)

WinPcap [39](#)

X

XDR functionality [4](#)

Z

zero-day vulnerabilities [243](#)

zero-trust network access (ZTNA) [131](#)

zero-trust security model [131](#)



packtpub.com

Subscribe to our online digital library for full access to over 7,000 books and videos, as well as industry leading tools to help you plan your personal development and advance your career. For more information, please visit our website.

Why subscribe?

- Spend less time learning and more time coding with practical eBooks and Videos from over 4,000 industry professionals
- Improve your learning with Skill Plans built especially for you
- Get a free eBook or video every month
- Fully searchable for easy access to vital information
- Copy and paste, print, and bookmark content

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at packtpub.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at customercare@packtpub.com for more details.

At www.packtpub.com, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on Packt books and eBooks.

Other Books You May Enjoy

If you enjoyed this book, you may be interested in these other books by Packt:

The Packt logo, consisting of the word "packt" in a lowercase, sans-serif font enclosed in a red chevron-shaped bracket.**1ST EDITION**

Microsoft Intune Cookbook

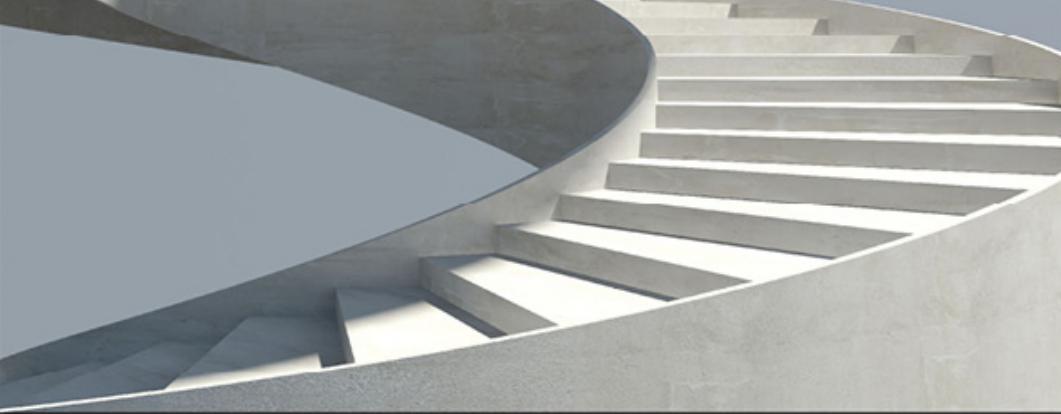
Over 75 recipes for configuring, managing, and automating your identities, apps, and endpoint devices

**ANDREW TAYLOR****Microsoft Intune Cookbook**

Andrew Taylor

ISBN: 978-1-80512-654-6

- Set up your Intune tenant and associated platform connections
- Create and deploy device policies to your organization's devices
- Find out how to package and deploy your applications
- Explore different ways to monitor and report on your environment
- Leverage PowerShell to automate your daily tasks
- Understand the underlying workings of the Microsoft Graph platform and how it interacts with Intune



packt

1ST EDITION

Mastering Microsoft 365 Defender

Implement Microsoft Defender for Endpoint, Identity,
Cloud Apps, and Office 365 and respond to threats



RU CAMPBELL | VIKTOR HEDBERG

Foreword by Heike Ritter, Principal Product Manager, Customer Experience, Microsoft Security

Mastering Microsoft 365 Defender

Ru Campbell, Viktor Hedberg

ISBN: 978-1-80324-170-8

- Understand the Threat Landscape for enterprises
- Effectively implement end-point security
- Manage identity and access management using Microsoft 365 defender
- Protect the productivity suite with Microsoft Defender for Office 365
- Hunting for threats using Microsoft 365 Defender

Packt is searching for authors like you

If you're interested in becoming an author for Packt, please visit authors.packtpub.com and apply today. We have worked with thousands of developers and tech professionals, just like you, to help them share their insight with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Share Your Thoughts

Now you've finished *Microsoft Defender for Identity in Depth*, we'd love to hear your thoughts! If you purchased the book from Amazon, please [click here to go straight to the Amazon review page](https://www.amazon.com/review/create.html?asin=B0B1XHJZLW&addtocart=1) for this book and share your feedback or leave a review on the site that you purchased it from.

Your review is important to us and the tech community and will help us make sure we're delivering excellent quality content.

Download a free PDF copy of this book

Thanks for purchasing this book!

Do you like to read on the go but are unable to carry your print books everywhere?

Is your eBook purchase not compatible with the device of your choice?

Don't worry, now with every Packt book you get a DRM-free PDF version of that book at no cost.

Read anywhere, any place, on any device. Search, copy, and paste code from your favorite technical books directly into your application.

The perks don't stop there, you can get exclusive access to discounts, newsletters, and great free content in your inbox daily

Follow these simple steps to get the benefits:

1. Scan the QR code or visit the link below



<https://packt.link/free-ebook/9781835884485>

2. Submit your proof of purchase
3. That's it! We'll send your free PDF and other benefits to your email directly

Contents

1. [Microsoft Defender for Identity in Depth](#)
2. [Foreword](#)
3. [Contributors](#)
4. [About the author](#)
5. [About the reviewers](#)
6. [Preface](#)
 1. [Who this book is for](#)
 2. [What this book covers](#)
 3. [To get the most out of this book](#)
 4. [Download the example code files](#)
 5. [Conventions used](#)
 6. [Get in touch](#)
 7. [Share Your Thoughts](#)
 8. [Download a free PDF copy of this book](#)
7. [Part 1: Mastering the Fundamentals of Microsoft Defender for Identity](#)
8. [Chapter 1: Introduction to Microsoft Defender for Identity](#)
 1. [The growing threat landscape and the role of MDI in ITDR](#)
 2. [Modern identity threats and strategic defense frameworks](#)
 1. [The Cyber Kill Chain](#)
 2. [MITRE ATT&CK framework](#)
 3. [The Unified Kill Chain](#)
 3. [MDI's strategic position in the cybersecurity ecosystem](#)
 4. [Unpacking key features and benefits of MDI](#)
 5. [Summary](#)
9. [Chapter 2: Setting up Microsoft Defender for Identity](#)
 1. [Technical requirements](#)
 2. [Pre-installation and planning checklist: laying the groundwork](#)
 1. [Licensing](#)
 2. [What permissions do you need?](#)
 3. [What are the operating system requirements?](#)
 4. [Other sensor requirements](#)
 5. [Networking](#)
 6. [PowerShell](#)
 7. [Data collection](#)

8. [User profiling](#)
 9. [Sizing](#)
 10. [Prerequisites for AD FS and AD CS](#)
 11. [Active Directory service accounts](#)
 3. [Deployment of MDI – a step-by-step guide](#)
 1. [Following with your own lab environment](#)
 2. [Getting the MDI installation package and access key](#)
 4. [Navigating step-by-step proxy configuration for MDI](#)
 1. [Installing TinyProxy](#)
 2. [Configuring TinyProxy](#)
 5. [Ensuring success with post-installation activities](#)
 1. [DSAs](#)
 2. [Configuring SAM-R](#)
 3. [Setting the gMSA in the Defender XDR portal](#)
 4. [Verifying the DSA](#)
 5. [Defender XDR unified RBAC](#)
 6. [Summary](#)
10. [Chapter 3: Leveraging MDI PowerShell for Automation and Management](#)
 1. [Technical requirements](#)
 2. [Primer on the MDI PowerShell module](#)
 1. [Installing the MDI PowerShell module](#)
 2. [Module file overview](#)
 3. [Understanding the module and its functions](#)
 3. [Crafting advanced PowerShell scripts for MDI management](#)
 1. [Health issues API](#)
 4. [Automation in action – case studies and scripting scenarios](#)
 1. [Monitoring the MDI service via Azure Monitor](#)
 2. [Monitoring the MDI configuration with Azure Monitor and custom alert rules](#)
 3. [Sending health issues and security alerts via syslog to Microsoft Sentinel](#)
 5. [Summary](#)
 11. [Part 2: Advanced Configuration, Integration, and Threat Detection](#)
 12. [Chapter 4: Integrating MDI with AD FS, AD CS, and Entra Connect](#)
 1. [Technical requirements](#)
 2. [Integrating MDI with AD FS](#)

1. [How AD FS authentication works](#)
 2. [Configuring AD FS for MDI sensor installation](#)
 3. [Validating the AD FS integration](#)
 3. [Integrating MDI with AD CS](#)
 1. [How AD CS works](#)
 2. [Importance of MDI on Certificate Servers](#)
 3. [Configuring AD CS for MDI sensor installation](#)
 4. [Validating the AD CS integration](#)
 4. [Integrating MDI with Entra Connect](#)
 1. [How Entra Connect works](#)
 2. [Configuring Entra Connect for the MDI sensor](#)
 3. [Validating the Entra Connect integration](#)
 5. [Expanding MDI across multiple Active Directory forests](#)
 1. [The concept of multiple forests](#)
 2. [Types of trusts in multi-forest environments](#)
 3. [Prerequisites for MDI in a multi-forest environment](#)
 6. [VPN integration – securing remote activities and data flow](#)
 1. [Understanding RRAS and RADIUS](#)
 2. [Configuring Microsoft RRAS](#)
 7. [Summary](#)
13. [Chapter 5: Extending MDI Capabilities Through APIs](#)
1. [Technical requirements](#)
 2. [Introduction to the MDI API](#)
 1. [Getting started with Microsoft Graph API](#)
 3. [Building custom integrations and automations](#)
 1. [Identifying integration opportunities](#)
 2. [Type of use cases](#)
 4. [Summary](#)
14. [Chapter 6: Mastering KQL for Advanced Threat Detection in MDI](#)
1. [Technical requirements](#)
 2. [KQL for beginners – querying MDI data](#)
 1. [The history of KQL and its ecosystem](#)
 2. [Understanding your MDI data](#)
 3. [Getting started with KQL](#)
 4. [Practical tips for effective queries](#)
 5. [Hunting tables in MDI](#)
 6. [Practical use of hunting tables](#)

3. [Advanced KQL techniques for deep threat detection](#)
 1. [Understanding attack paths in AD](#)
 2. [MDI and the attacker's kill chain](#)
 3. [Crafting KQL queries for threat detection](#)
 4. [Real-world case studies – detecting advanced attacks with KQL](#)
 1. [Prerequisites](#)
 2. [PtH attack](#)
 3. [Kerberoasting](#)
 4. [DCShadow attack](#)
 5. [Summary](#)
 6. [Further reading](#)
15. [Part 3: Operational Excellence with Microsoft Defender for Identity](#)
 16. [Chapter 7: Investigating and Responding to Security Alerts](#)
 1. [Developing a methodical approach to alert investigation](#)
 1. [Understanding the MDI alert system](#)
 2. [User Entity](#)
 3. [Lateral movement paths \(LMPs\)](#)
 4. [Initial triage and categorization](#)
 5. [Root cause analysis](#)
 2. [Real-world playbook – responding to advanced threats](#)
 1. [Defining advanced threats](#)
 2. [Pre-incident preparation](#)
 3. [Incident detection and validation](#)
 4. [Response strategy and execution](#)
 3. [Incident response – an action plan for high-stakes situations](#)
 1. [Building an incident response team](#)
 2. [Incident Response Plan \(IRP\)](#)
 4. [Summary](#)
 17. [Chapter 8: Utilizing MDI Action Accounts Effectively](#)
 1. [Technical requirements](#)
 2. [Configuring and securing action accounts](#)
 1. [Understanding action accounts – what are they and why do they matter?](#)
 2. [Best practices for action account configuration – getting it right the first time](#)
 3. [Security measures – protecting your action accounts from compromise](#)

3. Real-world scenarios and use cases
 1. Automated threat response – leveraging action accounts for quick reactions
 2. Case study – detecting and responding to credential theft and lateral movement
 3. Operational efficiency – how action accounts streamline security processes
 4. Summary
18. Chapter 9: Building a Resilient Identity Threat Detection and Response Framework
 1. Technical requirements
 2. Designing proactive threat-hunting strategies with MDI
 1. Understanding the threat-hunting methodology
 2. The importance of logging and accurate detection
 3. Developing security use cases
 4. Leveraging behavioral analytics and MDI in hypothesis-driven hunting
 3. Elevating your ITDR posture – Continuous improvement with MDI
 1. Learning from total identity compromise incidents
 2. Implementing identity-hardening strategies
 4. Disaster recovery and incident response – preparing for the inevitable
 1. Establishing an incident response plan
 2. Automating responses to identity-based incidents with SOAR
 3. Disaster recovery for identity systems
 5. Summary
19. Chapter 10: Navigating Challenges: MDI Troubleshooting and Optimization
 1. Diagnosing common MDI issues
 1. Spotting the signs of trouble
 2. Using tools and logs to find problems
 2. Configuration and connectivity fixes
 1. Checking key configuration settings
 2. Removing a malfunctioning MDI sensor manually
 3. Network connectivity troubleshooting
 3. Resolving security alert misfires
 1. Customizing detection rules and applying filtering techniques

2. [Adjusting alert settings for better accuracy](#)
 4. [Operational guide](#)
 1. [Daily tasks](#)
 2. [Weekly tasks](#)
 3. [Monthly tasks](#)
 4. [Quarterly/ad-hoc tasks](#)
 5. [Summary](#)
 6. [Future reading](#)
20. [Index](#)
 1. [Why subscribe?](#)
 21. [Other Books You May Enjoy](#)
 1. [Packt is searching for authors like you](#)
 2. [Share Your Thoughts](#)
 3. [Download a free PDF copy of this book](#)

Landmarks

1. [Cover](#)
2. [Table of Contents](#)
3. [Index](#)