Microsoft Security Workshop: Implementing PowerShell Security Best Practices

Course outline

Module 1: PowerShell Fundamentals

Introduced in 2006, Windows PowerShell is a scripting language, a command-line shell, and a scripting platform built on Microsoft .NET Framework. Despite the scripting designation, Windows PowerShell features a range of characteristics common for programming languages, including its object-oriented nature, extensibility, C#-like syntax, and the ability to interact directly with .NET classes, their properties, and methods. The primary objective of Windows PowerShell was to help IT professionals and power users control and automate the administration of the Windows operating system and applications that run on Windows. With the introduction of .NET Core in 2016, Microsoft extended the scope of PowerShell to other operating system platforms, leading to an open-source, GitHub-hosted project, named PowerShell Core. You can use PowerShell Core on macOS 10.12, a variety of 64-bit Linux distributions, in addition to the 32-bit and 64-bit Windows operating system, including Windows 10 running on Advanced Reduced Instruction Set Computing Machine (ARM) devices.

In this module, you will learn about PowerShell fundamentals, including its architectural design, its editions and versions, and basics of interacting with PowerShell.

Lessons

- Overview of Windows PowerShell
- PowerShell editions and versions
- Running PowerShell

After completing this module, you will be able to:

- Provide an overview of Windows PowerShell
- Describe PowerShell editions and versions
- Install and use Windows PowerShell and PowerShell Core

Module 2: PowerShell Operational Security

To take advantage of the benefits that Windows PowerShell has to offer, while at the same time, minimize security-related risks, it is essential to understand the primary aspects of Windows PowerShell operational security. In this module, you will learn about enhancing operating system security by leveraging built-in Windows PowerShell features and technologies that are part of the Windows PowerShell operational environment. Another aspect that is critical to consider in the context of this module is the role of Windows PowerShell in security exploits. According to empirical data, in majority of cases, Windows PowerShell is used as a post-exploitation tool. This implies that, at the point where a Windows PowerShell session is launched, an attacker already gained access to the security context in which the target system or the target user operates. This is the type of scenario that this module will focus on. In this case, Windows PowerShell serves as powerful and extremely flexible engine for executing arbitrary tasks on the local and remote computers, which,

incidentally, is the same reason that made Windows PowerShell extremely popular among system administrators.

There are obviously other types of attacks which rely on Windows PowerShell to gain unauthorized access to a target system. In this type of scenario, Windows PowerShell serves as an exploitation tool. We will explore these types of attacks in the last module of this course.

Lessons

- Managing Local Script Execution
- Managing remote execution capabilities of Windows PowerShell
- Managing remote execution capabilities of PowerShell Core
- Language Mode

After completing this module, you will be able to:

- Manage execution of local PowerShell scripts
- Manage remote execution of Windows PowerShell
- Manage remote execution of PowerShell Core
- Describe security implications of using Constrained Language Mode

Module 3: Implementing PowerShell-based Security

In the previous module, you learned about a number of security-related features built into Windows PowerShell and technologies that are part of the Windows PowerShell operational environment that help you with their enforcement. The purpose of this module is to present the most common and effective methods of leveraging Windows PowerShell to enhance operating system security. These methods include: > Protecting from unintended configuration changes by relying on PowerShell Desired State Configuration (DSC) > Implementing the principle of least privilege in remote administration scenarios by using Just Enough Administration (JEA) > Tracking and auditing events that might indicate exploit attempts by using Windows PowerShell logging

Lessons

- Windows PowerShell DSC
- Just Enough Administration (JEA)
- Windows PowerShell Auditing and Logging

After completing this module, you will be able to:

- Describe the architecture and components of Windows PowerShell DSC
- Implement JEA
- Recommend Windows PowerShell auditing and logging configuration

Module 4: Windows PowerShell-based Exploits and their Mitigation

Organizations cannot comprehensively identify gaps in security detection and response by solely focusing on breach prevention strategies. Understanding how to not only protect but also to detect

and respond to breaches is just as important "if not more so" than taking action to prevent a breach from occurring in the first place. By planning for the worst-case scenarios through Red Teaming (real-world attack and penetration), organizations can develop the necessary capabilities to detect attempted exploits and significantly improve responses associated with security breaches.

Red Teaming has become one of the most essential parts of developing and securing Microsoft's platforms and services. The Red Team takes on the role of sophisticated adversaries and allows Microsoft to validate and improve security, strengthen defenses and drive greater effectiveness of the entire security program. Red Teams enable Microsoft to test breach detection and response as well as accurately measure readiness and impacts of real-world attacks.

The purpose of the Blue Team is looking for creative and reliable defenses to detect and foil attacks orchestrated by the Red Team. The Blue Team is comprised of either a dedicated set of security responders or members from across the security incident response,

Engineering and Operations organizations. Regardless of their make-up, they are independent and operate separately from the Red Team. The Blue Team follows established security processes and uses the latest tools and technologies to detect and respond to attacks and penetration.

In this module, we will first approach the Windows PowerShell-based security from the Red Team's perspective. We will explore the most common Windows PowerShell-based techniques employed by hackers in order to leverage existing access to a Windows operating system to facilitate installation of malicious software, carry out reconnaissance tasks, establish its persistency on the target computer, and promote lateral movement. We will also review some of Windows PowerShell-based security tools that facilitate penetration testing, forensics, and reverse engineering of Windows PowerShell exploits. To conclude the module and the course, we will provide a summary of technologies recommended by the Blue Team that are geared towards implementing comprehensive, defense-in-depth security against Windows PowerShell-based attacks.

There are many documented exploits that utilize Windows PowerShell capabilities to carry out attacks that either target security flaws present in unpatched or out-of-date systems or to laterally expand the scope of such attacks once a single system is compromised. Note that the overview of such exploits presented in this module is not meant to be exhaustive. Our intention is to illustrate common patterns that such exploits follow and highlight the importance of a comprehensive defense in-depth strategy.

Lessons

- Windows PowerShell-based attacks
- Windows PowerShell-based security tools
- Summary of Windows PowerShell security-related technologies

Lab : Implementing Windows PowerShell Security

- Implement Windows PowerShell Logging by using DSC
- Carry out a Windows PowerShell-based exploit
- Implement Just Enough Administration

After completing this module, you will be able to:

- Provide examples of Windows PowerShell-based attacks
- Use Windows PowerShell-based security tools
- Provide an overview of Windows PowerShell-based security-related technologies
- Implement Windows PowerShell logging by using Desired State Configuration (DSC)
- Identify and mitigate Windows PowerShell-based exploits
- Implement Just Enough Administration (JEA)