Federal Office
for Information Security

Configuration Recommendations for Hardening of Windows 10 Using Built-in Functionalities

Version: 1.0

Federal Office for Information Security
Post Box 20 03 63
D-53133 Bonn
Phone: +49 22899 9582-0
E-Mail: bsi@bsi.bund.de
Internet: https://www.bsi.bund.de

Table of Contents

Figures

Tables

1 Introduction

1.1 Executive Summary

This document outlines the result of work package 11 of the project “SiSyPHuS Win10: Studie zu Systemaufbau, Protokollierung, Härtung und Sicherheitsfunktionen in Windows 10” (orig., ger.). This project is being conducted by the company ERNW Enno Rey Netzwerke GmbH on behalf of the German Federal Office for Information Security (orig. ger.: Bundesamt für Sicherheit in der Informationstechnik (BSI)).

The objective of this work package is to create a comprehensive hardening concept for the configuration of components of Windows 10. As required by the Federal Office for Information Security, Windows 10 LTSC 2019, 64 Bit in German language is the focus of this document.

2 General Concepts

Building on the results obtained in the work packages 2 to 10 a configuration recommendation for the hardening of Windows 10 has been created which covers the following use cases: “normal protection needs domain member” (orig. ger.: “normaler Schutzbedarf Domänenmitglied”, ND), “increased protection needs domain member” (orig. ger.: “hoher Schutzbedarf Domänenmitglied”, HD) and “normal protection needs standalone computer” (orig. ger.: “normaler Schutzbedarf Einzelrechner”, NE).

The recommendation is aimed at advanced users and administrators and is suitable for directly implementing the configuration settings of the operating system.

2.1 Scope

This document and the configuration recommendations it contains are valid for the Microsoft Windows 10 Long Term Servicing Channel (LTSC) operating system, version 2019. The Semi-Annual Channel (SAC) version equivalent to this is Windows 10, version 1809. It is functionally identical to Windows 10 LTSC version 2019 both in terms of the kernel and components which are included in both versions. Because LTSC versions are designed to maintain a consistent feature set and stability, Microsoft does not provide post-release feature upgrades and components that could be added with new functionality have been removed. The most important missing components are the Edge browser, the virtual assistant Cortana as well as all preinstalled Universal Windows Apps (“Store Apps”) including the Microsoft Store.

The recommendations given are based on the scenario of securing a standard office workstation with the available built-in functionalities of the operating system. This should also result in as few functional restrictions as possible. Depending on the usage scenario and purpose of the system (e.g., administrative activities, protection requirements of the processed information etc.), additional measures may have to be taken or stricter configuration settings implemented, whereby the recommendations provided here can serve as a baseline.

2.2 Scope Conditions

The configuration recommendations described in the following are based on the analysis conducted during the project, on security best practices, as well as on expertise by ERNW. All recommendations are both compared against the Center for Internet Security (CIS) Benchmark (cis_win10_1809, 2019) for Windows 10 Enterprise (Version 1809) as a globally known and widely adopted standard and the recommendations of the Security Baseline for Windows 10 1809 (ms_sec_bl_1809, 2021) by Microsoft. Deviations from the Security Baseline or the CIS Benchmark are explained and substantiated for the affected settings within this document (or in the attached list of configuration recommendations). Where settings do not deviate, a reference is made to the relevant section in the CIS benchmark or to the Security Baseline to help in finding the setting inside the other publications.

While creating this document, decisions for specific hardening recommendations were led by the following basic principles for increasing system security:

- Preventing known and widespread attack scenarios that, based on current knowledge, are being actively exploited or have a high probability of being exploited.
- Reducing the attack surface by disabling not needed (or outdated) functions and components.
- Improving data protection by disabling functions and components that rely on cloud services.
- Improving data protection by preventing unnecessary communication to the vendor as far as possible.
- Minimizing key security and privacy decisions as well as choices by the user
- Enforcing of reasonable default settings to prevent modifications by the user.

It is important to note that the recommended hardening settings should not be adopted in a productive environment without extensive preliminary testing as some settings may involve general or even very specific (depending on the use case) functional limitations. Furthermore, not all configuration parameters of the operating system are covered by the recommendations given in this document but only the configuration parameters that are relevant to improve the security of the operating system and fulfill the aforementioned basic principles.

Logging related configuration recommendations can be found in the document “Configuration Recommendations for Windows 10 Logging” (orig. ger.: “Empfehlung zur Konfiguration der Protokollierung in Windows 10”) of the SiSyPHuS project (work package 10).

The corresponding Group Policy objects for the recommendations regarding logging (work package 10) and hardening (work package 11) are provided as part of work package 12.

2.3 Definition of the Use Cases

This section defines the use cases considered in the recommendation and the potential impact on operating system functionalities.

2.3.1 Normal Protection Needs - Domain Member (ND)

The hardening measures / configuration recommendations are suitable for protecting the IT system from untargeted attacks and infections with widespread malware. In addition, the implementation of the hardening measures does not result in any significant restrictions on the use of functionalities of the IT system. The functionalities taken into account are listed under “Impact on Functionalities of the Operating System” (see section 2.4).

2.3.2 Increased Protection Needs - Domain Member (HD)

The hardening measures / configuration recommendations are suitable for protecting the IT system from targeted attacks with customized malware. Restrictions on the use of functionalities of the IT system are acceptable. The functionalities taken into account are listed under “Impact on Functionalities of the Operating System” (see section 2.4).

Note: The implementation of (HD) requires that the settings of (ND) as well as the ones for (HD) are configured.

2.3.3 Normal Protection Needs - Standalone Computer (NE)

The hardening measures / configuration recommendations are suitable for protecting the IT system from untargeted attacks and infections with widespread malware. In addition, the implementation of the hardening measures does not result in any significant restrictions on the use of functionalities of the IT system. The functionalities taken into account are listed under “Impact on Functionalities of the Operating System” (see section 2.4).

2.4 Impact on Functionalities of the Operating System

For every recommended configuration the observable impact on functionalities of the operating system and common applications is described in keywords. Especially the impact on the following functionalities has been considered:

- Working locally with office programs
- Network capability of the IT system
- Multimedia capability of the IT system

- Display of web pages and active content
- Remote maintenance access
- Enterprise deployment
- Active Directory integration of the IT system
- Execution of programs and applications
- Installation of further applications on the operating system

If no effects of the recommendation or deviations from the default behavior could be observed, the respective sub item belonging to the corresponding configuration recommendation is omitted.

3 General Recommendations

The following sections describe general recommendations for information security which should be considered for a secure operation of Windows 10 systems. As these recommendations are not necessarily implementable via technical configuration (with built-in Windows tools) or go beyond the defined scope of this document, these measures are only described in general terms.

3.1 Acquisition of Hardware and Software from Trusted Sources

Hardware as well as software can be manipulated and therefore contain malware and backdoors. To ensure the trustworthiness of an IT system, it should be ensured that all hardware and software in use come from a known and trusted source. In concrete terms this means:

- Hardware used should only be obtained from trustworthy sources, ideally via the vendor itself. This does not only apply to the client system itself but also for any additional hardware that will be connected to it (such as external input devices, external data storages, docking stations etc.).
- All software that is being installed should only be obtained from trustworthy sources. If possible, software should be obtained through a central deployment process of the company or organization. If this is not possible, commercial software should be acquired through a designated reseller (including online stores) or the vendor directly.
- If open-source software is installed or open-source code (such as scripts) is executed, the developer should be identified and software or scripts should be obtained from sources associated with the developer, e.g., the associated GitHub repository.
- If software or code in general is obtained and downloaded from online sources, the integrity of the data should be verified. This can be achieved by verifying a hash value provided by the manufacturer or by checking an existing digital signature regarding the integrity of the file and authenticity of the issuer.
- Software should, whenever possible, be transferred to the system over a secure and encrypted channel to prevent so-called man-in-the-middle attacks (in which an attacker modifies data during transfer) and to enable the possibility to authenticate the issuer, e.g., by using Transport Layer Security (TLS). This is especially important if the data is transferred over the Internet.

3.2 Separation of Standard User Accounts and Administrative Accounts

Typical tasks of a standard user (e.g., accessing web pages, receiving emails, editing documents) introduce a significantly higher risk that could lead to a compromise than tasks carried out by an administrator. To prevent an attacker from directly gaining extensive control over a system after successfully compromising a user account, a minimum of two different accounts should exist. One of these accounts should have no special privileges and is therefore not an attractive goal for an attacker, while the other account is an administrative account that is assigned the privileges necessary for administrative activities. The standard user account should be used for the initial login and tasks that do not require administrative privileges while the administrative account should only be used for tasks which require higher privileges. This reduces the likelihood of a successful compromise of an administrative account as it presents an additional hurdle for an attacker. This recommendation applies both to local accounts as well as to domain accounts in an Active Directory environment. Furthermore, accounts that are not in use or are not needed for the operation of the system should be deactivated or completely removed.

Windows supports the use of separated accounts with different privileges through the functionality User Account Control (UAC) since actions requiring elevated privileges will automatically prompt for administrative user account credentials if the logged-in user does not already possess these.

3.3 Implementation of Secure Password Policies

An attacker may be able to compute (e.g., by breaking a hash value to obtain the associated plaintext password) or even guess (e.g., by so-called password spraying, where the same password is tested for successful authentication for all users in an environment) easy passwords with a password attack. A reasonably secure password policy enforces that only passwords can be selected that meet certain requirements which makes them significantly more difficult to be calculated (or guessed). In general, the following should be considered when choosing a password: the length is more decisive for the password quality than the complexity, and this additional length means that the password can be changed less often.

The longer a password is, the longer an attacker needs to calculate it, e.g., based on a password hash. Also, the likelihood that a password can be successfully guessed is reduced. Whereas increasing the password complexity for short length passwords only marginally increases the password quality (see (ms_pass, 2021)). If the password strength is sufficient, the password can be changed less often or not at all (see ORP.4.A23 in (bsi_it_gs_orp4, 2021)). However, if a password compromise is suspected, the password should be changed immediately in any case. Furthermore, passwords that are included in lists of known and common passwords should be avoided.

The configuration of the password policy recommended in this document refers primarily to the local configuration of a Windows system and therefore affects local user accounts. In an Active Directory environment, the password policy should be enforced on domain level for all systems and domain user via Group Policies. In addition, there should be a granular distinction between account types in an Active Directory environment to be able to impose a stricter password policy for administrative user accounts and technical service accounts (e.g., through so-called Fine-grained Password Policies).

3.4 Secure Password Storage

Passwords that are stored in clear text on the system (e.g., in text files, scripts, browsers) can be easily read and stolen by an attacker who possesses the permissions of the logged-in user. A password manager can be used to store passwords encrypted and protected from unauthorized access. Therefore it is recommended to use a password manager on the system. After installation, the user creates a password database and chooses a reasonably secure master password (in accordance with a previously defined password policy). All passwords that grant the user access to websites, shares, applications etc. should then be stored in the password database. These passwords should ideally be generated by the password manager and therefore unique. The password database and the passwords stored in it can be accessed by entering the master password.

3.5 No Password Reuse

If the same password is used for different purposes (e.g., applications, systems, websites), it can also be compromised in multiple places by an attacker. If the password is compromised in one of these places, the attacker can reuse the same password to access further independent services for which the same password is in use. A password manager does not only ensure a secure storage for multiple passwords (see section 3.4) but can also be used to generate and store random and unique passwords for each application / service. For every service that requires a password the user should therefore create a unique password and store it securely in the password manager. This recommendation also applies to different user accounts (e.g., domain user accounts) which are used by the same person.

3.6 Regular Updating of Firmware, Operating System, and Installed Applications

As an essential link between the hardware of a device and the operating system, the firmware is especially important. To ensure the security of the hardware platform on which the operating system and different applications will be installed, the firmware of the system should always be up-to-date. For this purpose, it is necessary to check in regular intervals whether a new update has been released by the manufacturer. The update process should also include a verification of the stored cryptographic key material and certificates respectively which define which firmware (updates) and operating system loaders can be run on the system. This mainly involves checking the information stored in the so-called Allowed Signature Database (db) and the Forbidden Signature Database (dbx), which are used for the secure boot process.

For the Windows operating system and Microsoft applications, Microsoft provides updates on a regular basis. Updates to patch critical security vulnerabilities are usually released outside of the regular update cycle. Therefore, all updates, patches and hotfixes for the operating system and Microsoft applications should be installed as timely as possible after their release via the Windows Update functionality of the operating system (concrete recommendations for the configuration of the Windows Update mechanism are given in this document). In Active Directory environments, a so-called Windows Server Update Service (WSUS) can also provide the updates in a centralized manner to be able to control which systems receive which updates.

Third-party software which is additionally installed on the system also increases the attack surface of the system. Like the operating system itself, installed applications may have unpatched security vulnerabilities that may lead to a compromise of the whole system if exploited. Standardized applications in a managed environment (such as browsers, email clients, document processing applications etc.) should be updated by a centralized patch management solution as soon as possible after patch release. On individual computers, manual checks should be carried out on a regular basis to determine whether new updates are available. Here, there are often no defined regular intervals in which updates are published, and update information is usually not standardized, which generally makes this process more difficult.

In general, the recommendations in section 3.1 should be followed for all updates (especially in the case of a manual update) to ensure the integrity and authenticity of the update files and respectively the source from which they were obtained.

3.7 Installation of Required Applications and Operating System Components Only

Each additional software component provides additional attack surface and must be updated (sometimes manually) on a regular basis (see section 3.6). Therefore, only applications and operating system components should be installed that are required for the user tasks conducted on the system or for the operation of the system itself. Preinstalled but not required applications and Windows components should be removed or if this is not possible at least deactivated (such as for Windows services) during the initial setup. A general recommendation for the deactivation of operating system components is being given in this document. However, which applications and functionalities are generally needed and should therefore be installed and activated, must be determined individually according to the intended use. Additionally, all installed applications should be reviewed regularly for their necessity and be removed if not required.

3.8 Use of Hard Disk Encryption

To ensure the confidentiality of data stored in the file system of the operating system and to prevent non-authorized modifications of data, a software solution for hard disk encryption should be used. This is especially relevant when the physical security of the device cannot be ensured, e.g., in the case of theft. Not only the operating system and data partitions should be encrypted, but ideally the complete hard disk.

Moreover, before the operating system is started, the encryption software should perform a user authentication (the so-called pre-boot authentication) to prevent parts of the cryptographic material used for decrypting the hard disk from being loaded into memory and potentially being read by an attacker. The encryption algorithm used should be chosen in such a way that it is not trivially possible for an attacker to decrypt the hard disk without knowledge of the encryption key. The encryption key should be randomly generated and stored securely. A backup and recovery concept should be designed bevor enabling hard disk encryption to prevent that the loss of the encryption key leads to a loss of all local data.

4 Configuration Recommendations

All recommendations for Group Policy settings were moved into a separate table type document to accommodate for the number of recommendations and preserve the readability and structure of the present document. Furthermore, presenting the configuration recommendations in tabular form increases their readability and usability. The complete list of all hardening recommendations based on Group Policy settings can be found in the document “Configuration Recommendations for Hardening of Windows 10 Using Built-in Functionalities: Group Policy Settings” (orig. ger.: “Konfigurationsempfehlungen zur Härtung von Windows 10 mit Bordmitteln: Gruppenrichtlinien-Einstellungen”). In addition, further setting recommendations exist that are considered in the following chapter.

5 Additional Configuration Recommendations

The following sections contain recommendations for configurations that cannot exclusively be configured via Group Policies and need additional explanations.

5.1 (HD) Windows Defender Application Control

Windows 10 implements a functionality called configurable code integrity (hereinafter referred to as Windows Defender Application Control – WDAC). WDAC restricts the execution of non-trustworthy code. "Non-trustworthy code" means programs which integrity and / or authenticity cannot be verified (e.g., because the program was changed or was downloaded from an unknown source). WDAC uses user-defined criteria to validate executable files, i.e., allow only certain files to be executed. The criteria can be based on file attributes (e.g., hash values, file names). The criteria are defined in so-called WDAC policy files. These files are first in XML (Extensible Markup Language) format and are then converted to binary files which are finally read and interpreted by WDAC.

WDAC is enabled by the Group Policy setting `Deploy Windows Defender Application Control` under the path `Computer Configuration\Administrative Templates\System\Device Guard`. Recommendations for the configuration of WDAC policies can be found in (ERNW_WP7), section 3.1.2, section 3.1.3, and section 3.1.4.

The following sections describe the elements to consider for the secure use of WDAC.

5.1.1 Signing of WDAC Policies

Ensure that every WDAC policy is digitally signed to prevent unauthorized modifications of policies.

In organizations, this requires a properly managed signing certificate, ideally provided by a public key infrastructure (PKI). The WDAC policy itself should be signed on a dedicated system to ensure the protection of the signing certificate and the signing process. Furthermore, the signed WDAC policy should only be transferred via a secure channel to the corresponding target systems.

The configuration interfaces for the signing of WDAC policies are listed below.

5.1.1.1 Configuration Interface: PowerShell

Set-AuthenticodeSignature `Set-AuthenticodeSignature` is a PowerShell command with which binary files can be digitally signed.

```
Set-AuthenticodeSignature -FilePath $policy -Certificate $cert
```

- `FilePath` specifies the path of the policy file (in binary format)
- `Certificate` specifies the certificate object used for signing. This object can be created with for example the PowerShell cmdlet `Get-PfxCertificate` (ms_pfx, 2021)

Detailed information regarding the `Set-AuthenticodeSignature` PowerShell command can be found here (ms_ps, 2021).

5.1.1.2 Configuration Interface: Windows-Applications

SignTool.exe The `SignTool.exe` application is a command line tool with which, among others, binary files can be digitally signed. `SignTool.exe` is part of the Software Development Kit (SDK).

```
SignTool.exe sign /v /fd sha256 /f $cert /p $pass $policy
```

- `sign` specifies that a binary file should be signed
- `/f` specifies the file path of the certificate used for signing (in PFX format)
- `$policy` specifies the path of the policy file (in binary format)
- `/fd` specifies the file hashing algorithm for creating file signatures
- `/v` specifies that detailed and comprehensive output and warning messages should be displayed
- `/p` specifies the password used to decrypt the signing certificate

Detailed information regarding the `SignTool.exe` can be found here (ms_sign, 2021).

5.1.2 Requirements for the Secure Use of WDAC

Unified Extensible Firmware Interface (UEFI)

Ensure that the UEFI firmware is activated.

The UEFI firmware provides a secure storage for relevant WDAC configuration parameters and files, e.g., for the protection of the integrity of a WDAC policy file. Storing sensitive WDAC configuration parameters and files in the firmware allows for a protection against manipulation by unauthorized Windows users.

5.2 (HD, ND, NE) Virtualization Based Security

Virtualization Based Security (VBS) is a Microsoft Hypervisor-based functionality which separates the traditional Windows architecture into two isolated environments (herein after called the secure kernel mode and the normal mode). Separating the Windows architecture in a secure kernel and a normal mode allows for the isolation of security critical functionality from the normal mode which leads to an increased protection from unauthorized access. This includes, for example, securely storing and managing Windows credentials or verifying Windows applications to be executed. It should be emphasized that the use of VBS to protect certain functionalities makes a forensic analysis of these functions (e.g., analysis of the memory or debugging) a challenging or unfeasible task.

VBS is conceptually divided into two components. The VBS core component which is provided by the Microsoft Hypervisor and the so called VBS applications, such as “Virtualization Based Protection of Code Integrity” and “Credential Guard”.

VBS is enabled by the Group Policy setting `Turn on Virtualization Based Security` under the path `Computer Configuration\Administrative Templates\System\Device Guard`.

The following sections describe the recommended configurations for the secure use of the VBS core component and the VBS applications.

5.2.1 VBS Core Component

The following VBS core component settings increase the security of the Windows 10 system initialization and protect against platform attacks (e.g., attacks that abuse hardware-level functions).

5.2.1.1 Secure Boot and DMA Protection

Ensure that the following Group Policy setting `Turn on Virtualization Based Security` under the path `Computer Configuration\Administrative Templates\System\Device Guard` is set to: `Enabled`, and • `Select Platform Security Level` is set to: `Secure Boot and DMA Protection`.

Default configuration: Not Configured

Secure Boot is a standard for booting computers securely, so that the computer only loads software that the computer manufacturer and / or the operating system vendor deems trustworthy. `Secure Boot and DMA Protection` should be enabled on all platforms that support Direct Memory Access (DMA). These are platforms with devices with input-output memory management units (IOMMUs). This policy option enables mechanisms to defend against DMA-based attacks and requires hardware support.

5.2.1.2 Secure Launch Configuration

Ensure that the following Group Policy setting `Turn on Virtualization Based Security` under the path `Computer Configuration\Administrative Templates\System\Device Guard` is set to: `Enabled`, and

- `Secure Launch Configuration` is set to: `Enabled`.

Default configuration: Not Configured

This configuration ensures that a platform is only launched with trustworthy code. It activates the dynamic root of trust (DRTM) functionality which protects the platform from firmware-based attacks.

5.2.2 VBS Applications

The following configurations of settings of the VBS applications increase the security of the Windows user logon process and the integrity checking mechanism.

5.2.2.1 Credential Guard

Ensure that the following Group Policy setting `Turn on Virtualization Based Security` under the path `Computer Configuration\Administrative Templates\System\Device Guard` is set to: `Enabled`, and

- `Credential Guard Configuration` is set to: `Enabled with UEFI lock`.

Default configuration: Not Configured

Credential Guard uses VBS to securely store and manage Windows credentials. The configuration of `Enabled with UEFI lock` enables Credential Guard and instructs Windows to store relevant Credential Guard configuration parameters in UEFI's secure storage.

5.2.2.2 Virtualization Based Protection of Code Integrity

Ensure that the following Group Policy setting `Turn on Virtualization Based Security` under the path `Computer Configuration\Administrative Templates\System\Device Guard` is set to: `Enabled`, and

- `Virtualization Based Protection of Code Integrity` is set to: `Enabled with UEFI lock`

The option `UEFI Memory Attributes Table` is not selected.

Default configuration: Not Configured

Virtualization based protection of code integrity uses VBS to securely verify Windows applications to be executed. The configuration of `Enabled with UEFI lock` enables virtualization based protection of code integrity and instructs Windows to store relevant virtualization based protection of code integrity configuration parameters in UEFI's secure storage.

5.2.3 Requirements for the Secure Use of VBS

Unified Extensible Firmware Interface (UEFI)

Ensure that the UEFI firmware is enabled.

The UEFI firmware provides secure storage for relevant VBS configuration parameters. Storing relevant VBS configuration parameters in the firmware provides stronger protection from tampering by unauthorized Windows users.

Trusted Platform Module (TPM)

Ensure that TPM is present and enabled.

The TPM provides a secure storage for security-critical data such as cryptographic material.

5.3 (HD, ND, NE) Trusted Platform Module

The Trusted Platform Module (TPM) is an electronic chip that provides basic security features at hardware level. The TPM is protected against tampering by hardware and can be used as a secure storage for data or to protect the integrity of a system. Furthermore, the chip provides a mechanism to securely generate and manage cryptographic material.

The TPM is a component that accepts and processes transmitted commands (called TPM commands). Some TPM commands and functions are protected so that they can only be executed by authorized users. User authorization is done by specifying an authorization value. One such value is the *owner authorization value* (`OwnerAuth`), which is the central authentication feature for managing the TPM.

The configuration recommendations that should be considered for the secure use of the TPM are described in the following sections.

5.3.1 Blocking Commands

Ensure that the following Group Policy setting `Ignore the default list of blocked TPM commands` under the path `Computer Configuration\Administrative Templates\System\Trusted Platform Module Services` is set to: `Disabled`.

Default configuration: Not Configured

The policy configures Windows access control to be able to restrict the execution of TPM commands. The `Disabled` value configures the default command blocking and instructs Windows to allow only certain TPM commands.

5.3.2 Standard User Lockout Duration and Standard User Lockout Threshold (individual/total)

Ensure that the following Group Policy settings: `Standard User Lockout Duration` under the path `Computer Configuration\Administrative Templates\System\Trusted Platform Module Services` is set to: `Enabled`, and • `Duration for counting TPM authorization failures (minutes)` is set to: `30`; `Standard User Individual Lockout Threshold` and `Standard User Total Lockout Threshold` under the path `Computer Configuration\Administrative Templates\System\Trusted Platform Module Services` is set to: `Enabled`, and

- `Maximum number of authorization failures per duration` is set to: `5`.

Default configuration: Not Configured

The TPM can limit authorization attempts to prevent brute force attacks. The TPM counts the number of TPM authorization failures within a time period (`Standard User Lockout Duration, Duration for counting TPM authorization failures`) and locks itself if a certain threshold (`Standard User Lockout Threshold)` is reached. If a maximum number of incorrect authorization attempts has been exceeded (`Maximum number of authorization failures per duration`), the TPM goes into lockout mode. In lockout mode, no more commands requiring authorization can be processed by the TPM.

The recommended configuration values have proven to be operationally feasible while providing a reasonable security benefit.

5.3.3 Requirements for the Secure Use of the TPM

Windows compliant TPM initialization

Execute the PowerShell command `Enable-TpmAutoProvisioning`.

The TPM should be automatically initialized by Windows to generate a secure `OwnerAuth` automatically and to function properly. The automatic TPM initialization by Windows is called `auto provisioning`. The PowerShell command `Enable-TpmAutoProvisioning` enables the automatic TPM initialization.

5.4 (HD, ND, NE) Windows-Telemetry

Windows Telemetry is a Windows 10 component responsible to collect and transfer automatically data to a Microsoft operated backend infrastructure. Since the nature and extent of the collected data, the security of the data transmission and of the data storage and processing in the telemetry backend are not fully known, Windows Telemetry should be disabled.

5.4.1 Deactivation of the Telemetry Service and ETW Sessions

Ensure that the telemetry service and the corresponding ETW sessions are disabled.

For both enterprise environments and end users alike, considering typical operational requirements, disabling the telemetry service (`Connected User Experiences and Telemetry`) and corresponding ETW sessions (storage areas where logged events are written) provide the best balance of effectively disabling telemetry and minimizing the operational impact.

The relevant ETW providers (entities that log events) write their data into the ETW sessions `Autologger-DiagTrack-Listener` and `DiagTrack-Listener`. These sessions are the source of the telemetry data. Disabling these sessions will stop the telemetry data collection. To disable both sessions and the transmission of telemetry data, the `Connected User Experiences and Telemetry` service must first be disabled. This prevents the `DiagTrack-Listener` session from being initiated. Additionally, the `Autologger-DiagTrack-Listener` session must be disabled. This can be achieved by setting the value of the corresponding Registry key to "0".

For environments with increased protection needs (HD), the recommendations for network-based mitigation measures should also be taken into account and implemented (see section 3.2 in (ERNW_WP4_1)). A preferential measure would be the fitting configuration of a DNS resolver, as these are present in most corporate environments as well as in environments of well-versed end users and effectively fulfill the filtering requirements to prevent telemetry communication at the network level with little effort.

5.4.1.1 Disabling Connected User Experiences and Telemetry

Ensure that in the Group Policy settings the service `Connected User Experiences and Telemetry (DiagTrack)` under the path `Computer Configuration/Windows Settings/Security Settings/System Services` is set to: `Disabled`.

Default configuration: Automatic

By disabling the service `Connected User Experiences and Telemetry` the initialization of the `DiagTrack-Listener` session, which is a source for part of the telemetry data, and the transmission of logged data to the telemetry backend are prevented.

Alternatively, the service can be disabled using one of the following procedures:

Interface	Path/Command
Services (services.msc)	`Connected User Experiences and Telemetry` → `Properties` → `Startup type` → `Disabled`
Registry Editor	`HKLM\SYSTEM\CurrentControlSet\Services\DiagTrack\ Start = 4`
PowerShell	`Set-ItemProperty -Path HKLM:\SYSTEM\CurrentControlSet\Services\DiagTrack \ -Name Start -Value 4`

Table 1 Deactivation of Connected User Experiences and Telemetry

5.4.1.2 Disabling Autologger-Diagtrack-Listener

Ensure that the Registry value `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\WMI\Autologger\AutoLogger-DiagTrack-Listener\Start` is set to `0`.

Default configuration: 1

The `Autologger-Diagtrack-Listener` session is initialized early in the boot process and stores its logged data in the local file system in binary form. After the telemetry service is started, the service processes this data. To prevent the logging and local storage of this data, the `Autologger-Diagtrack-Listener` session should be disabled.

Alternatively, the session can be disabled using one of the following procedures:

Interface	Path/Command
PowerShell	`Set-ItemProperty -Path HKLM:\SYSTEM\CurrentControlSet\Control\WMI\Autologger\AutoLogger-Diagtrack-Listener\ -Name Start –Value 0`

Interface	Path/Command

Performance Monitor (perfmon.exe)	`Data Collector Sets ->`
	`Start Event Trace Sessions ->`
	`AutoLogger-Diagtrack-Listener ->`
	`Properties ->`
	`Trace Session ->`
	Remove checkmark from `Enabled`

Table 2 Deactivation of Autologger Diagtrack Listener

Detailed descriptions of the different configuration options for telemetry-relevant settings can be found in work package 4.1 of the SiSyPHuS project (ERNW_WP4_1)

5.5 PowerShell and Windows Script Host

5.5.1 PowerShell

The Windows PowerShell provides a powerful administrative environment. A variety of operating system resources can be accessed through Windows PowerShell. These include, for example, the Windows Registry, the Windows certificate store, Windows environment variables, the file system, and several PowerShell resources. This enormous variety of functions not only provides the administrator with many tools, but PowerShell is also increasingly being used by sophisticated attackers to carry out complex attacks.

5.5.1.1 (HD, ND, NE) Disabling PowerShell Version 2.0

Ensure that PowerShell version 2.0 is disabled.

Unlike PowerShell version 5.1 (and up), PowerShell version 2.0 does not implement relevant security functionalities, such as support for the anti-malware scan interface (AMSI) or PowerShell script block logging (ERNW_WP8). Therefore, PowerShell version 2.0 is considered a security risk and should be disabled.

5.5.1.1.1 Configuration Interface: PowerShell

Disable-WindowsOptionalFeature `Disable-WindowsOptionalFeature` is a PowerShell command which disables PowerShell version 2.0.

```
Disable-WindowsOptionalFeature -Online -FeatureName
MicrosoftWindowsPowerShellV2Root
```

5.5.1.1.2 Configuration Interface: Windows Features

Windows Features is a Windows service for enabling or disabling system components.

Open the Windows application `Control Panel`. Click on `Programs and Features`. Click on `Turn Windows-Features on or off`. In the dialog box `Windows Features` disable the option `Windows PowerShell 2.0`. Apply the changes with `OK`.

5.5.1.2 (HD, ND, NE) Restricting PowerShell Script Execution

Ensure that a PowerShell execution policy is set that is appropriate for the environment in question.

PowerShell supports so called execution policies. Execution policies can be used to configure under which conditions PowerShell allows the execution of scripts. These policies differ in their strictness and are suitable for different scenarios where there is a relatively high or normal need for protection. It is important

to emphasize that the execution policies prevent the accidental execution of scripts. However, this does not prevent an attacker from, for example, copying the contents of a script into a PowerShell instance and thus bypassing a set execution policy (ms_ep, 2021).

In order to find the individual compromise between security and usability for each environment, it is useful to categorize or classify the execution policies. Table 3 provides an overview of the different execution policies and the scenarios in which they are suitable (increased protection needs - H; normal protection needs – N).

Execution Policy	Description	Use Case
`Restricted`	This execution policy prohibits the execution of scripts.	H
`AllSigned`	This execution policy allows the execution of scripts that are digitally signed by a trustworthy issuer. The execution of other scripts is prohibited.	
`RemoteSigned`	This execution policy allows the execution of scripts that originate from the computer itself, whereas scripts that originate from the Internet must be signed by a trusted issuer. Execution of all other scripts is not allowed.	N
`Unrestricted`	This execution policy allows the execution of all scripts.	

Table 3 PowerShell Execution Policies

In sections 5.5.1.2.1 and 5.5.1.2.2 an overview is given about the interfaces which allow the configuration of the above listed execution policies.

5.5.1.2.1 Configuration Interface: PowerShell

Set-ExecutionPolicy `Set-ExecutionPolicy` is a PowerShell command that allows for the setting of the PowerShell execution policy.

`Set-ExecutionPolicy -ExecutionPolicy <Policy> -Scope <Scope>`

- `<Policy>` specifies one of the execution policies listed in Table 3 in the column “Execution Policy” (see (ms_expol, 2021))
- `<Scope>` specifies the scope of the effect of the execution policy, such as:
  - `UserPolicy`: the execution policy affects the current user
  - `Process`: the execution policy affects the current PowerShell session
  - `LocalMachine`: the execution policy affects all users on the computer on which the execution policy is configured

Detailed information regarding the `Set-ExecutionPolicy` PowerShell command can be found here (ms_expol, 2021).

5.5.1.2.2 Configuration Interface: Group Policy

Execution policies can be configured with the Group Policy setting `Turn on script execution` under the path `Computer Configuration\Administrative Templates\Windows PowerShell`.

Default Configuration: Not configured (the execution of scripts is not allowed, which corresponds to the execution policy `Restricted` –see Table 3).

Configurable values:

- `Allow only signed scripts:` is equivalent to the execution policy `AllSigned` – see Table 3

- `Allow local scripts and remote signed scripts:` is equivalent to the execution policy `RemoteSigned` – see Table 3
- `Allow all scripts`: is equivalent to the execution policy `Unrestricted` – see Table 3.

5.5.1.3 (HD) Restricting PowerShell Scripting Language (Local Computer)

PowerShell supports the concept of language modes. A language mode determines the allowed PowerShell commands and language elements that a user can execute. The language mode is a PowerShell session variable and therefore session-specific (ERNW_WP8). Table 4 PowerShell Language Modes

gives an overview over all PowerShell language modes.

PowerShell-Language Mode	Description
`FullLanguage`	This language mode permits all commands and language elements.
`RestrictedLanguage`	This language mode permits all commands; however, it restricts the execution of script blocks. The use of some variables and operators is allowed. Script elements such as assignment statements and method calls are not allowed.
`NoLanguage`	This language mode permits all commands; however, it restricts all language elements.
`ConstrainedLanguage`	This language mode permits all commands and language elements; however, it restricts commands and elements based on data types. A complete documentation of the functions of this language mode, can be found here (ms_lm, 2021).

Table 4 PowerShell Language Modes

In local PowerShell sessions the default language mode is `FullLanguage`. Windows only enforces the `ConstrainedLanguage` language mode if a system-wide application control solution like Device Guard User Mode Code Integrity (UMCI) is enabled. If UMCI is enabled, PowerShell scripts that are not specified in the active UMCI policy file are executed in `ConstrainedLanguage` language mode. The scripts specified in the provided WDAC policy are executed in `FullLanguage` language mode. Therefore, only trusted PowerShell code can be executed in `FullLanguage` language mode.

Due to the high restrictions of the `ConstrainedLanguage` language mode and the UMCI policies, this language mode is only suitable for use cases with a very high need for protection.

5.5.1.4 Secure Use of PowerShell Remoting

PowerShell remoting is a PowerShell functionality which allows users to access PowerShell over a network connection remotely. Although this functionality is convenient for administrative purposes, it may also be abused by attackers.

Ensure that PowerShell remoting is only enabled when required.

In some Windows versions (e.g., Windows Server 2012) PowerShell remoting is enabled by default. PowerShell remoting can be enabled with the PowerShell command `Enable-PSRemoting` (ms_er, 2021).

5.5.1.4.1 Disabling PowerShell Remoting

Disabling PowerShell remoting consists of several steps. This section focuses on PowerShell Version 5.1.

1. Execute the PowerShell command `Disable-PSRemoting`.

This PowerShell command disables remote access to all PowerShell session configurations. In section 5.5.1.4.2 all PowerShell session configurations are described.

2. (Option 1) Execute the PowerShell commands `Stop-Service WinRM -PassThru` and `Set-Service WinRM -StartupType Disabled -PassThru`.

These PowerShell commands stop and disable the `WinRM`-Windows service. This service provides remote access to PowerShell over a network connection. In Windows, this service is required not only for PowerShell remoting, but also for server administration.

If the `WinRM` service cannot be disabled because of practical reasons, its PowerShell listeners (network endpoints that enable PowerShell remoting) should be disabled.

2. (Option 2) Execute the PowerShell commands `dir wsman:\localhost\listener` and `Remove-Item -Path WSMan:\localhost\listener\<Listener>`.

The output of the PowerShell command `dir wsman:\localhost\listener` are the names of all listeners of the `WinRM` services. The PowerShell command disables a particular listener (`<Listener>` specifies the name of a listener).

3. Execute the PowerShell command `Set-NetFirewallRule -DisplayName 'Windows Remote Management (HTTP-In)' -Enabled False -PassThru | Select -Property DisplayName, Profile, Enabled`.

When PowerShell remoting is enabled by the PowerShell command `Enable-PSRemoting`, firewall exceptions are created to allow remote access to TCP port 5895. This command disables these firewall exceptions.

4. (optional, only on standalone systems) Ensure that the Registry value `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\LocalAccountTokenFilterPolicy` is set to: `0`.

This setting ensures that users cannot run remote commands with an administrator access token without triggering a UAC (User Account Control) prompt. When PowerShell remoting is enabled with the PowerShell command `Enable-PSRemoting`, users can run remote commands with an administrator access token without triggering an UAC prompt. This setting does not affect domain users.

5.5.1.4.2 Restricting PowerShell Remoting

If PowerShell remoting is to be enabled, it should be restricted for security reasons. Just Enough Administration (JEA) is a PowerShell security mechanism that implements the least privilege principle. It creates PowerShell instances (PowerShell sessions) to administrate systems with a limited set of available PowerShell functionalities. These are the minimum functionalities required to perform certain administrative tasks on a system.

JEA uses PowerShell remoting so that users can connect to a PowerShell session with limited functionality. This session is called the JEA endpoint. Users can connect to the JEA endpoint by using the PowerShell `Enter-PSSession` command (ms_es, 2021).

A typical approach to configure JEA consists of several phases. The following paragraphs explain these phases and provide a minimal example of a JEA configuration. For detailed information about JEA see (ms_jea, 2021).

Phase 1: Planning The configuration and deployment of JEA requires careful planning. The goal of the first phase is to find the right balance between functionality and security for a given scenario. To this end, two questions should be answered:

- Which users (or user groups) should have access to the JEA endpoint?

- What PowerShell functionality should be made available to these users?

For example, the PowerShell instance of the user `test` should be restricted to allow only the running of the `Get-Help` and `Get-History` PowerShell commands.

Phase 2: Specify JEA Roles In the second phase the allowed PowerShell functionalities for a particular user (or user group) in the form of JEA roles are specified. JEA roles are specified in JEA role capability files (file name extension: `psrc`).

A role capability file should be stored in a folder named `RoleCapabilities`. This folder should be stored in a PowerShell module folder, for example, in a subfolder stored in `C:\Program Files\WindowsPowerShell\Modules`.

Important: Access to the `RoleCapabilities` folder should be restricted. Only trusted administrators should be able to access the folder and modify the role capability file. Otherwise, an untrusted user could modify the role capability file and bypass the restrictions imposed by JEA.

Create and edit a role capability file.

The PowerShell command `New-PSRoleCapabilityFile` creates an empty role capability file template. The created file can be edited with a text editor. In the role capability file, the PowerShell functionalities identified in phase 1 should be specified. Figure 1 shows the role capability file for the user `test` (Filename: `testRole.psrc`).

```
@{

GUID = '0712c3ff-41cc-4dd8-9368-8e6fad3f90f1'

VisibleCmdlets = 'Get-Help', 'Get-History'

}
```

Figure 1 Role capability file for the user test (testRole.psrc)

JEA supports the specification of permitted PowerShell functionalities by keywords that are written to the role capability file. Some of these keywords are:

- `VisibleCmdlets`: specifies the permitted PowerShell commands (cmdlets, see Figure 1 Role capability file for the user test (testRole.psrc), (ERNW_WP8));
- `VisibleProviders`: specifies the permitted PowerShell provider, e.g., `Registry`. PowerShell uses providers to access system resources;
- `VisibleExternalCommands`: specifies the executable files stored in the file system that are permitted to be executed through PowerShell.

For detailed information on role capability files, see. (ms_rfd, 2021).

Store the created role capability file.

When the role capability file is completed, it should be copied to a folder named `RoleCapabilities`. The PowerShell commands in Figure 2 Storing the role capability file testRole.psrc: [1] create a folder for a PowerShell module named `testJEA` (`C:\Program Files\WindowsPowerShell\Modules\testJEA`); [2] register the module; [3] create the `RoleCapabilities` folder (as a subfolder of `testJEA`) and copy the `testRole.psrc` file from the current folder to this folder.

```
$modulePath = Join-Path $env:ProgramFiles "WindowsPowerShell\Modules\testJEA"
New-Item -ItemType Directory -Path $modulePath

New-Item -ItemType File -Path (Join-Path $modulePath "testJEAFunctions.psm1")
New-ModuleManifest -Path (Join-Path $modulePath "testJEA.psd1") -RootModule "testJEAFunctions.psm1"

$rcFolder = Join-Path $modulePath "RoleCapabilities"
New-Item -ItemType Directory $rcFolder
Copy-Item -Path .\testRole.psrc -Destination $rcFolder
```

Figure 2 Storing the role capability file testRole.psrc

Phase 3: Set session configuration The third phase involves the mapping between users and JEA roles. In addition, settings for the PowerShell session constrained by JEA (i.e., JEA endpoint) can be specified. These settings are specified in a session configuration file (file name extension: `pssc`).

Create and edit a session configuration file.

The PowerShell `New-PSSessionConfigurationFile` command creates an empty session configuration file template. This file can be edited with a text editor. Figure 3 Session configuration file for the user test (testSession.pssc) depicts the session configuration file for the user `test` (file name: `testSession.pssc`).

```
@{

SchemaVersion = '2.0.0.0'

GUID = 'a73e1139-562e-459b-a3b1-63b2facfcf0e'

SessionType = 'RestrictedRemoteServer'

TranscriptDirectory = 'C:\Transcripts\'

RunAsVirtualAccount = $true

RoleDefinitions = @{ 'DESKTOP-ASRI2Q0\test' = @{ RoleCapabilities = 'testRole' } }

}
```

Figure 3 Session configuration file for the user test (testSession.pssc)

JEA supports the specification of session settings by keywords which are written in the session configuration file. Some of these keywords are:

- `RoleDefinitions`: the mapping between users (or user groups) and JEA roles (see Figure 3). The key word `RoleCapabilities` should specify the name of the role capability file without the file name extension (`testRole` in Figure 3);
- `RunAsVirtualAccount`: specifies that the user is allowed to use the JEA approved PowerShell functionality with administrative privileges (see Figure 3);

`LanguageMode`: specifies a PowerShell language mode (see Table 4Table 4 PowerShell Language Modes

-);
- `SessionType`: specifies a session type. In the context ofJEA the `RestrictedRemoteServer` session type is recommended (see Figure 3). This session type configures the session to allow only the PowerShell functionalities specified by the `RoleDefinitions` keyword. In addition, the PowerShell language mode `NoLanguage` is enforced;

- `TranscriptDirectory`: instructs PowerShell to store a complete log of all entered PowerShell commands to a file in the specified folder. This is recommended.

Detailed information regarding session configuration files can be found here (ms_skd, 2021).

Register the session configuration.

When the session configuration file is completed, the session configuration should be registered under a specific name. A session configuration can be registered using the PowerShell command `Register-PSSessionConfiguration`. For example, the PowerShell command:

```
Register-PSSessionConfiguration -Name testJEAEndpoint -Path
C:\ProgramData\JEAConfigurations\testSession.pssc -Force
```

registers the session configuration file `testSession.pssc` (see Figure 3) which is stored in the folder `C:\ProgramData\JEAConfigurations` under the name `testJEAEndpoint`.

Phase 4: Testing When the session configuration is registered, the JEA endpoint is enabled and should be tested.

Test the JEA endpoint by using the `Enter-PSSession` PowerShell command.

Example: The PowerShell command `Enter-PSSession testhost -ConfigurationName testJEAEndpoint -Credential test` connects to the JEA endpoint hosted on the computer named `testhost` and specifies the session configuration named `testJEAEndpoint` (parameter `ConfigurationName`). The PowerShell session is set up for the user `test` (parameter `Credential`). As specified in the role capability file, the user `test` can execute only the `Get-Help` and `Get-History` PowerShell commands. All other PowerShell functionalities are not allowed (see Figure 1 Role capability file for the user test (testRole.psrc)).

5.5.2 Windows Script Host

The Windows Script Host (WSH) provides a runtime environment for a number of scripting languages (ERNW_WP8). The Windows Script Host can be used by users and administrators to automate tasks.

5.5.2.1 (ND, NE) Execution of Trustworthy Scripts

Ensure that the Registry value `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows Script Host\Settings\TrustPolicy` is set to `2`.

This configuration prevents the execution of unsigned scripts, i.e., scripts with digital signatures that cannot be verified or have been signed by untrusted issuers.

By default, the execution of all scripts is allowed. If the Registry value `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows Script Host\Settings\TrustPolicy` does not exist, it should be created.

Unsigned scripts, for example, can then no longer be used for administration.

5.5.2.2 (HD) Disabling the Windows Script Host

Ensure that the Registry value `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows Script Host\Settings\Enabled` is set to `0`.

This configuration disables the Windows Script Host. By default, the Windows Script Host is enabled. If the Registry value `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows Script Host\Settings\Enabled` does not exist, it should be created.

If the Windows Script Host is disabled, scripts for which execution is supported by WSH can no longer be executed. This protects against the execution of malicious scripts, but also prevents legitimate scripts from running.

It is important to emphasize that this setting only blocks the execution of scripts through the user interface. The WSH core functionalities are still active.

5.5.2.3 (HD) Disabling Windows Script Host Remoting

Ensure that the Registry value `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows Script Host\Settings\Remote` is set to `0`.

This configuration disables WSH Remoting. By default, WSH Remoting is enabled. If WSH Remoting is enabled, WSH scripts can be executed on a Windows system that is accessible via a network connection. This way, attackers can execute malicious scripts on these Windows systems.

5.6 (HD, ND, NE) Firmware

Firmware is software used to control the hardware of a computer. The current firmware standard is UEFI (Unified Extensible Firmware Interface) and the older firmware standard is BIOS (basic input/output system). UEFI provides more functionalities than BIOS, including security-relevant settings. This section focuses on the secure configuration of the UEFI firmware.

Because there are multiple firmware manufacturers and some firmware settings are hardware-dependent, some settings may be missing or some of the settings listed in this section may not be available on all platforms. This section focuses on the most commonly available firmware settings.

5.6.1 Administrator Password

Ensure that a UEFI administrator password is set.

The UEFI administrator password prevents unauthorized modifications to UEFI settings. UEFI can usually be configured to require this password every time the system boots or (only) when the UEFI graphical user interface is started.

5.6.2 Restricting the Boot Order

Ensure that the list of devices from which an operating system can be booted (i.e., the boot order) is limited to only those devices that are required.

This setting prevents the booting of operating systems from unknown devices. Such booting can lead to data theft if the hard disk is not encrypted.

5.6.3 Disabling Legacy Firmware Functionalities

Ensure that all legacy firmware functionalities are disabled.

This setting disables the use of legacy firmware functionalities (BIOS functionalities), such as legacy boot mode or legacy boot options. Legacy firmware functionalities do not provide the security related functionalities offered by UEFI (e.g., secure boot, see section 5.2.1.1).

5.6.4 Secure Firmware Updates

Ensure that only firmware updates signed by the manufacturer can be installed.

This setting prevents the installation of firmware updates from an unknown source.

Ensure that firmware rollback is disabled or at least protected by the administrator password (i.e., locked), see section 5.6.1.

This setting prevents the installation of an older UEFI firmware version.

Ensure that the firmware updates via Windows are enabled.

This setting ensures the automated and trustworthy installation of the latest firmware version.

5.6.5 Miscellaneous

UEFI is a technical requirement for several security-related Windows components and functionalities. This section lists some UEFI settings that must be enabled for these components and functionalities to work properly.

Ensure that TPM is enabled.

This setting enables the TPM (see section 5.3). If it is not enabled in the firmware, Windows cannot use the TPM.

Ensure that the CPU virtualization extensions are enabled.

This setting enables the CPU virtualization extensions (e.g., Intel Virtualization Technology - VT-x). If they are not enabled, the virtualization-based security features are not functional (see section 5.2).

Ensure that secure boot is enabled.

This setting enables secure boot. If it is not enabled in the firmware, the secure boot functionality will not secure the Windows boot process.

Ensure that execution prevention is enabled.

This setting enables execution prevention (NX). Execution prevention prevents unauthorized access to certain memory areas. Therefore, it protects against the execution of malicious software.

Appendix

Tools Used

Tool	Availability and Description
Group Policy Editor	*Availability:* Included with Windows 10 *Description:* A tool for configuring Group Policy settings
Registry Editor	*Availability:* Included with Windows 10 *Description:* A tool for configuring the Registry
Services	*Availability:* Included with Windows 10 *Description:* A tool for configuring system services
Performance Monitor	*Availability:* Included with Windows 10 *Description:* A tool for displaying and configuring performance data and logs

Reference Documentation

bsi_it_gs_orp4. (2021, March 11). Retrieved from https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/Kompendium_Einzel_PDFs_2021/02_ORP_Organisation_und_Personal/ORP_4_Identitaets_und_Berechtigungsmanagement_Editon_2021.pdf?__blob=publicationFile&v=2

cis_win10_1809. (2019, November 22). *CIS Microsoft Windows 10 Enterprise (Release 1809) Benchmark v1.6.1*. Retrieved from https://www.cisecurity.org/cis-benchmarks/

ERNW_WP10. (n.d.). SiSyPHuS Win10 (Studie zu Systemaufbau, Protokollierung, Härtung und Sicherheitsfunktionen in Windows 10): Work Package 10.

ERNW_WP11. (n.d.). SiSyPHuS Win10 (Studie zu Systemaufbau, Protokollierung, Härtung und Sicherheitsfunktionen in Windows 10): Work Package 11.

ERNW_WP12. (n.d.). SiSyPHuS Win10 (Studie zu Systemaufbau, Protokollierung, Härtung und Sicherheitsfunktionen in Windows 10): Work Package 12.

ERNW_WP2. (n.d.). SiSyPHuS Win10 (Studie zu Systemaufbau, Protokollierung, Härtung und Sicherheitsfunktionen in Windows 10): Work Package 2.

ERNW_WP4_1. (n.d.). SiSyPHuS Win10 (Studie zu Systemaufbau, Protokollierung, Härtung und Sicherheitsfunktionen in Windows 10): Work Package 4.1.

ERNW_WP5. (n.d.). SiSyPHuS Win10 (Studie zu Systemaufbau, Protokollierung, Härtung und Sicherheitsfunktionen in Windows 10): Work Package 5.

ERNW_WP7. (n.d.). SiSyPHuS Win10 (Studie zu Systemaufbau, Protokollierung, Härtung und Sicherheitsfunktionen in Windows 10): Work Package 7.

ERNW_WP8. (n.d.). SiSyPHuS Win10 (Studie zu Systemaufbau, Protokollierung, Härtung und Sicherheitsfunktionen in Windows 10): Work Package 8.

ms_applocker. (2020, Juli 17). Retrieved from https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-defender-application-control/applocker/applocker-overview

ms_ep. (2021, March 11). Retrieved from https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_execution_policies?view=powershell-5.1

ms_er. (2021, March 11). Retrieved from https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/enable-psremoting?view=powershell-5.1

ms_es. (2021, March 11). Retrieved from https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/enter-pssession?view=powershell-5.1

ms_expol. (2021, March 11). Retrieved from https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.security/set-executionpolicy?view=powershell-5.1

ms_jea. (2021, March 11). Retrieved from https://docs.microsoft.com/en-us/powershell/scripting/learn/remoting/jea/overview?view=powershell-5.1

ms_lm. (2021, March 11). Retrieved from https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_language_modes?view=powershell-5.1

ms_mss. (2020, Juli 17). Retrieved from https://blogs.technet.microsoft.com/secguide/2016/10/02/the-mss-settings/

ms_pass. (2021, March 11). Retrieved Juli 17, 2020, from https://docs.microsoft.com/en-us/archive/blogs/msftcam/password-complexity-versus-password-entropy

ms_pfx. (2021, March 11). Retrieved from https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.security/get-pfxcertificate?view=powershell-5.1

ms_ps. (2021, March 11). Retrieved from https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.security/set-authenticodesignature?view=powershell-5.1

ms_rfd. (2021, March 11). Retrieved from https://docs.microsoft.com/en-us/powershell/scripting/learn/remoting/jea/role-capabilities?view=powershell-5.1

ms_sec_bl_1809. (2021, March 11). Retrieved from https://www.microsoft.com/en-us/download/details.aspx?id=55319

ms_sign. (2021, March 11). Retrieved from https://docs.microsoft.com/en-us/dotnet/framework/tools/signtool-exe
ms_skd. (2021, March 11). Retrieved from https://docs.microsoft.com/en-us/powershell/scripting/learn/remoting/jea/session-configurations?view=powershell-5.1

Abbreviations

AMSI: anti-malware scan interface 20
BIOS: basic input/output system 27
DMA: Direct Memory Access 15, 16
DNS: Domain Name System 18
ETW: Event Tracing for Windows 18
JEA: Just Enough Administration 23, 24, 25, 26
LTSC: Long-Term Servicing Channel 5, 6
TPM: Trusted Platform Module 17, 18, 28
UAC: User Account Control 23
UEFI: Unified Extensible Firmware Interface 15, 16, 17, 27, 28
UMCI: User Mode Code Integrity 22
VBS: Virtualization Based Security 15, 16, 17
WDAC: Windows Defender Application Control 14, 15, 22
WSH: Windows Script Host 26, 27