Evaluation of Persistence Methods Used by Malware on Microsoft Windows Systems

Amélie Dieterich, Matthias Schopp, Lars Stiemert, Christoph Steininger and Daniela Pöhn[a]
Universität der Bundeswehr München, Neubiberg, Germany

Keywords: Malware, Persistence, Evaluation, Windows Security.

Abstract: The usage of persistence methods has become common, as adversaries seek to remain undetected with their malware on systems for longer periods. This raises the question of how effective frequently used persistence methods are across different versions of the Microsoft Windows operating system. To answer this question, a metric is developed by which persistence methods can be quantitatively evaluated and compared. The metric is subsequently applied to eight persistence mechanisms across four different Microsoft Windows operating systems. In our results, there is no difference in the performance of methods between operating systems and a majority of mechanisms scored similarly overall. There is, however, a significant decline in performance when defensive mechanisms are enabled. The results emphasize the effectiveness of basic persistence methods of Microsoft Windows operating systems.

1 INTRODUCTION

Malware persistence has become commonplace, from backdoors to well-known ransomware such as WanaCry. Thereby, adversaries receive persistent access to computers. On the other hand, it becomes more difficult to detect and remove them. Other examples of the importance of persistence are Advanced Persistent Threats (APTs). APTs are cyber attacks on a computer network that remain undetected for an extended period of time. They are often launched by state-funded actors with the aim of sabotaging or spying on the confidential data of companies or authorities. With persistent malware becoming ubiquitous, it is important to be able to gauge how dangerous a persistence method is, in order to understand the threat posed by specific methods. This can be achieved by scoring persistence methods. Currently, most classifications of persistence methods focus solely on similarities between how persistence is achieved such as with (MITRE, 2022) ATT&CK. ATT&CK summarizes adversary techniques and tactics into an openly available knowledge base with one category being dedicated to persistence tactics. Here, methods are categorized according to which medium is abused to gain persistence.

In order to assess the danger emerging persistence methods pose, a baseline is needed against which newly established methods can be compared. Microsoft Windows is the most widely used desktop operating system (OS) with a market share of 74.99% as of November 2022 (StatCounter, 2022). In consequence, we concentrate on Microsoft Windows OS with the recent versions of Windows 10, Windows 11, Windows Server 2019, and Windows Server 2022. Since defensive tools, such as anti-virus software, are regularly updated to maintain their effectiveness, malware persistence methods may be rendered ineffective. As a result, it is important to consider whether persistence methods are detected and/or prevented by defensive software on a system. Consequently, we focus on the following research questions:

R1: How effective are common persistence methods on recent Microsoft Windows OSs?

R2: To what extent do the built-in defenses of Windows 11 and Windows Server 2022 protect against persistence methods compared to their respective predecessors?

The contribution of this paper is twofold: 1) we present a systematic evaluation approach including a metric, and 2) we present the results of a practical evaluation of persistence methods on several different Windows OSs.

The paper is organized as follows: Section 2 provides an overview of related work. Next, in Section 3,

[a] https://orcid.org/0000-0002-6373-3637

Dieterich, A., Schopp, M., Stiemert, L., Steininger, C. and Pöhn, D.
Evaluation of Persistence Methods Used by Malware on Microsoft Windows Systems.
DOI: 10.5220/0011710200003405
In *Proceedings of the 9th International Conference on Information Systems Security and Privacy (ICISSP 2023)*, pages 552-559
ISBN: 978-989-758-624-8; ISSN: 2184-4356

common persistence methods for Microsoft Windows OSs are explained. The methodology of our research is outlined in Section 4, while its application within the practical evaluation is described in Section 5. The results are summarized and discussed in Section 6. Section 7 concludes the paper and provides an outlook for future work.

2 RELATED WORK

While (MITRE, 2022; Hunkeler, 2022) describe common persistence methods, research approaches either analyze persistence methods or propose ways to detect them.

(Gittins and Soltys, 2020) focus on persistence techniques in well-known malware, such as Emotet, TrickBot, and OceanLotus, and their identification through malware analysis. The authors show that even complex malware often gains persistence through well-known methods. (Barr-Smith et al.,) analyze the Living-off-the-Land (LotL) technique with several malware samples, showing that it is often used at APTs. (Rana et al., 2021) examine various persistence methods in Microsoft Windows OS. Even though the authors analyze several methods, it stays unclear which exact versions of Microsoft Windows OS and test setup they use and how often the single steps are repeated.

(Mankin, 2013) applies DIONE, an infrastructure for rule-based disk input/output (I/O) monitoring and analysis to detect if a persistence method is installed and if the mechanism is successful. Thereby, the author demonstrates the effectiveness of classifying persistence mechanisms according to their capabilities. (Dubey, 2019) combines static and dynamic analysis to identify malware persistence mechanisms. The persistence methods are detected by identifying various functions and files that were modified by the malware. The author concludes that a combination of static, dynamic, advanced static, and advanced dynamic is most suitable for identifying malware persistence techniques. In contrast, (Laurenza et al., 2020) propose a group of one-class classifiers to detect APT samples. Similarly, (Reischaga et al., 2020) use heuristic, static, and dynamic analysis for detecting, assessing, and measuring malware threats. To understand the impact of adversarial examples on malware detection, (Park and Yener, 2020) review practical attacks against malware classifiers.

Summary: The focus of the described related work is on analyzing malware regarding their persistence methods as well as understanding these mechanisms. However, few approaches rate the effectiveness in terms of the threat posed by different persistence techniques and compare them to each other.

3 PERSISTENCE METHODS

This section explains a selected number of methods to maintain persistence within a compromised system. As the practical evaluation in Section 5 focuses solely on current versions of the Microsoft Windows OSs, only methods that operate on Windows are considered in the following. Furthermore, methods covered in this section have been commonly used by adversaries in well-known malware, such as Mimikatz, Agent Tesla, and BabyShark (MITRE, 2022). We exclude pre-OS persistence methods such as bootkits because these are more commonly used in APTs.

3.1 User Account

Abusing existing user accounts to gain initial access to a system is a common tactic. In addition, this method can be used for persistence. In the following, we describe the three categories of valid accounts, account manipulation, and account creation, which work for both local and domain accounts.

Valid Account: If attackers obtain valid credentials to an existing account, they are able to abuse these to maintain their foothold within a system. This is referred to as valid account persistence. Intruders obtain account credentials through a wide variety of tools and techniques, such as credential leaks, social engineering, password cracking, and key-loggers (Thomas et al., 2017). The persistence gained through a valid account is not affected by changes (updates, upgrades) to the OS. However, the credentials can be altered and accounts can be deleted. This locks the intruder out, although installed malware may remain active. Detecting the abuse of accounts is challenging because it concerns identifying unusual behavior. Indicators to identify account abuse exist, for instance, logons outside of business hours and one account logged into multiple machines simultaneously. Inactive accounts are especially vulnerable to attacks, as there is no user present to identify abnormal activity on the account (CISA, 2022).

Account Manipulation: To prevent the loss of access caused by the users changing their password, intruders may change the credentials themselves. This type of approach is referred to as account manipulation. However, it is not restricted to only password modification. Other tactics include any action by which an intruder preserves access to a compromised account, for example, by modifying permission groups or al-

lowing remote access to accounts. The drawback of modifying user accounts lies in the higher probability of detection as these actions are likely to show up in the logs. Furthermore, a changed password on an active account will be noticed by the user. The privileges required for a specific type of modification vary: Changing an account's password is possible with limited rights, however, modifying permission groups requires administrator privileges.
Account Creation: Lastly, persistence can be achieved by adding new accounts to a compromised system. While the persistence is similar to the categories described before, this tactic offers benefits in terms of detection since the account has no additional users. Therefore, there is no other user, who might recognize unusual activity on the account. However, creating a new account will show up in the logs and requires administrator privileges.

3.2 Scheduled Task/Job

Persistence achieved by abusing built-in functions for scheduling tasks falls into the category of scheduled task/job persistence. On Windows, this type of persistence is often achieved through the Windows Task Scheduler (WTS). The tasks performed by WTS run in background sessions. Triggers determine when a task with its actions, i. e., items or jobs, is to be executed. The security context in which a task is run is defined by principals. Settings are used to determine how a task is to be run with respect to conditions external to the task. Registration information contains administrative information that is gathered when a task is created. The data component is supplied by the author and contains additional documentation about the task (Microsoft, 2021).

The WTS can be abused for persistence through the creation of tasks with the `schtasks` utility resp. graphical user interface (GUI) that regularly execute malicious code. Tasks can only be created by administrators but executed with any privileges. Regular user activity does not interfere with the completion of tasks. Furthermore, as tasks are designed to be automatically executable during boot-up, scheduled tasks have no difficulty surviving system reboots. When updating or upgrading a Windows OS, all data will be transferred. However, it is not uncommon for data to be lost during the procedure, possibly affecting the scheduled task (Potter and Nieh, 2005). Scheduled task persistence can be detected by monitoring for newly created scheduled tasks. Furthermore, changed entries related to scheduled tasks in `%systemroot%\System32\Tasks` may also indicate a malicious presence.

3.3 System Process

In Windows, system-level processes are referred to as Windows Services and are typically used to perform background system functions. Services can be defined by creating an application that is subsequently installed as a service using `InstallUtil.exe` and edited with the Service Control Manager. For this type of persistence, the malware will often install itself as a service. The persistence is similar to the scheduled task persistence because services are also system processes that are executed, for example, at boot-up or logon. Creating services requires administrator privileges, however, they are executed under system privileges. Consequently, this technique can be used for privilege escalation. Abuse of Windows services can be detected by monitoring the creation and modification of Windows services resp. monitoring changes in the service registry entries and unusual processes.

3.4 Boot or Logon Auto-Start Execution

When an adversary automatically runs a program at boot or logon by configuring system settings, it is referred to as boot or logon auto-start execution. There are many mechanisms in OSs intended for executing programs at boot or logon, including the Windows registry and particular directories from which programs are automatically run. Due to the fact that auto-start programs often run with elevated privileges, they may also be used for privilege escalation.
Shortcut Modification: Creating and modifying shortcuts resp. shell links, which are used to run a program during system boot or user login, can be abused to gain persistence. Shortcuts can either be added, modified, or entirely replaced within the Windows startup folder, which is executed during user logon. Shortcuts can persist through anything up to and including updates and upgrades. The required as well as the effective privileges for this technique vary, depending on the way it is applied. Detection of shortcut persistence is accomplished by monitoring for newly created or modified `.lnk` files.
Registry Run Keys/Startup: The Windows registry is a system-defined database in which configuration data is stored by applications and system components. Its structure is hierarchical and resembles a tree format, in which each node is a key and each key can contain both subkeys and data entries in the form of values. Two registry run keys are created by default, which can be abused to register malware. The registry is a built-in system feature and, therefore, maintains its foothold but does not survive a re-installation.

Monitoring the creation and modification of keys in the registry is one way to detect this method.

4 METHODOLOGY

In this section, we describe the used methodology with the evaluation procedure (Section 4.1), methods (Section 4.2), applied metric (Section 4.3), and limitations (Section 4.4).

4.1 Evaluation Procedure

The evaluation consists of the followings steps:

1. **Creation & Required Permissions:** The method establishes its foothold and its required permissions are determined. As accounts can be given a variety of different privileges, labeling them as either user or administrator is difficult. In order to simplify the designation, the default rights accounts assigned on creation are not manipulated during the evaluation. Required permissions are determined by trying to execute a given method, first with a user and then with an administrator account. After this step, the method should be ready to establish its persistence at the next reboot or logon.
2. **Affirmation & Effective Permissions:** In this step, it is determined whether a method has established its persistence. The method may first need to be triggered by, for example, logon, in order to execute. Hence, determining whether a method has gained persistence varies depending on the mechanism. During this step, the effective permissions will be reviewed by determining the privileges a method uses to run itself.
3. **Persistence:** To determine the depth of persistence we perform an action for each level of persistence (User action: log off and on; reboot: restart; update: installing an OS update; re-installation: re-installing the OS) and review whether the method has maintained its foothold.
4. **Detection:** First, we observe anti-virus while executing a method. If the software triggers an alert or prevents the mechanism from executing, the method is considered detected. Then, we focus on monitoring. The logs are searched and the following are checked for new/modified entries: list of all accounts, list of scheduled tasks, entries in the two registry run keys, and both startup folders. Pre-OS changes are detected by monitoring for changes made on pre-OS boot mechanisms.
5. **Reliability:** The reliability is reviewed by repeatedly triggering the method to produce a reliable statement. For a reliable setting, logon/logoff-triggered methods are executed 50 times and reboot-triggered methods ten times.

4.2 Methods

Since we execute each method on different OSs, we define the following standardized approach for each:

Valid Accounts: Logon to user/admin account.

Account Manipulation: Password modification.

Account Creation: Create a new account and elevate privileges.

Scheduled Task: Create a task.

Windows Services: Create a new service.

Shortcut Modification: Create a shortcut inside the current user resp. common startup folder.

Registry Run Keys: Add entries to the two registry run keys.

4.3 Metric

The metric consists of the aspects of persistence, permissions, detection, and reliability. The points given to the different methods and categories are in accordance with the difficulty, desired outcome, and (MITRE, 2022).

Persistence: The methods are sorted into categories (user action, reboot, update, re-installation) based on what system changes they can withstand. User actions summarize everything a user might do on a daily basis. User actions are the most common of all actions to occur. Consequently, persistence methods that cannot maintain their foothold through these are the least useful. Points are awarded in descending order of frequency of occurrence. Re-installation (RI) of the OS is the least likely to occur in comparison to updates (UD), reboots (RB), and user actions (UA). Following, it is assigned 4 points, as shown in Table 1. If a method fulfills the requirements of multiple categories, it is to be placed in the highest.

Table 1: Valuation of persistence.

Persistence	UA	RB	UD	RI
Value	1	2	3	4

Permissions: Accounts can be categorized as the user, administrator, and system. For simplicity, we combine administrator and system. Administrator accounts have higher permissions than user accounts:

full control of the files, directories, services, and other resources on the local computer. Furthermore, they are capable of managing accounts, rights, and permissions. Users, on the other hand, are much more restricted in their rights. Overall, the lower the required permission is, the better for the attacker. Whereas for effective permission the opposite is true. As a result, the best case scenario is *user* $\rightarrow$ *admin*, and the worst case is *admin* $\rightarrow$ *user*. The distribution of points is accordingly and shown in Table 2.

Table 2: Valuation of required and effective permissions.

Req. vs Ef.	User	Admin
User	2	3
Admin	1	2

Detection: All mechanisms that are detected by pre-installed antivirus software such as the Windows Defender are contained in the category of Antivirus (AV). Monitoring (MO) is a wide category including all checks system administrators regularly perform, such as monitoring logs and important files. Methods that can only be detected by performing integrity checks on pre-OS boot mechanisms are assigned to Pre-OS (PO). All methods that cannot be assigned to any of the previous categories are counted as Not Viable (NV). Since the detection through antivirus software requires the least effort, it should be avoided by adversaries at all costs and, therefore, is assigned the lowest score. Pre-OS, on the other hand, requires noticeably more effort because these methods cannot be detected by host software-based defenses. The exact allocation of points can be found in Table 3.

Table 3: Valuation of detection.

Detection	AV	MO	PO	NV
Value	1	2	3	4

Reliability: Reliability is rated by the percentage of successful executions. Success is when a persistence mechanism has successfully gained persistence without any undesirable side effects. If a method is executed ten times, a probability below 50% indicates that fewer than five attempts are successful. This is undesirable (0 points), as a crash may alert the system administrator and renders the system unavailable to the adversary. In contrast, a probability of 96–100% gives an adversary a likely chance of success (4 points). Table 4 provides an overview of the points.

Table 4: Valuation of reliability.

Reliability	<50%	≥50%	>75%	>95%
Value	0	1	2	3

Final Score: In the final step, we combine the individual properties. For the sake of readability, a denotes the points achieved in Persistence, b in Permission, c in Detection, and d in Reliability. Considering that detection, permission, and reliability are secondary goals, persistence is multiplied by three to achieve a weighting of the primary goal. The final result P of a method is calculated as follows:

$$P = 3 \cdot a + b + c + d \qquad (1)$$

The maximum score that can be achieved is 22. A higher score indicates that a method is able to maintain its persistence for a longer period of time, compared to methods with a lower score.

4.4 Limitations

As described, we concentrate on the most often used methods. Nevertheless, further persistence methods exist and can be analyzed in future work. For the practical evaluation, a simple malware with bind-shell created by MSFvenom is used. More advanced malware, such as those utilized by APTs, can use more sophisticated methods, such as OS and application patching or supply-chain attacks. Hence, the results may differ. This may be evaluated in future work. In addition, we focus on recent Windows OS. Future versions of Windows and other OSs may result in different scores resp. require other methods.

5 PRACTICAL EVALUATION

This section describes the practical evaluation, first without and then with Windows Defender software.

5.1 Setup of the Evaluation

As indicated in Section 4.4, we generate the executable using MSFvenom. Whether the executable is runnable is detected through netstat by verifying the port is open. `bind.exe` is the malware being executed. It needs to be placed on the system before establishing persistence. For the evaluation, we will assume that the file is already on the system. The persistence is evaluated by the corresponding means. The account permissions are determined with `net user <username>`. The list of all accounts is retrieved utilizing `control.exe`. The task scheduler helps to identify scheduled tasks. Registry entries are seen by viewing the registry editor and files by the file explorer. To determine the reliability of logon/off, we force the system into a loop of logoffs and logons.

For the practical evaluation, we use virtual machines (VMs) on VMware Workstation 16 Pro 1. Between each method, the VM is reset to a base snapshot. All VMs have their network connection set to Network Address Translation (NAT), which means they share the IP address of the host system. For each OS, the first evaluation is without any defensive software, in contrast to the second and last evaluations. The defensive mechanisms include all security features that come preinstalled with each Windows OS. Furthermore, automatic updates are deactivated. The two accounts *User* and *Admin* are created on this system and the `bind.exe` is located in `C:\`.

5.2 Evaluation of Windows OSs

With the described methodology, we evaluate the following Windows OSs:

- Windows 10 Pro version 21H2 with KB4023057 and KB5013624 being the latest updates;
- Windows 11 Pro version 21H2 with KB5013943 and KB5013628 being the latest updates;
- Windows Server 2019 Standard Evaluation version 1809, OS-build 17763.253;
- Windows Server 2022 Standard Evaluation version 21H2, OS-build 20348.169.

The main difference is the stricter password policy for Windows Server. This though does not affect the implementation. All four OSs produce the same results. As consequence, we describe the practical evaluation for all four OS in the following.

Valid Accounts. For valid accounts, no previous access to the system is required and the gained permissions depend on the compromised account. In consequence, we will differentiate between two scenarios: in the first, we compromise a user account, and, in the second, we gain access to an account with admin privileges. Access to both accounts remains unchanged up to and including *Update*. However, a *Re-install* will cause the accounts to be removed. Log-on events are logged by default with the event ID 4624. Therefore, both scenarios are detected through *Monitoring*. The reliability of this method is dependent on the reliability of the OS. The reliability of Windows 10 is adequate, thereby both scenarios achieve 100% reliability. Table 5 provides an overview of the score for each scenario.

Table 5: Results of Valid Accounts.

Account Type	PS	PM	DE	RE	Score
User	3	2	2	3	16
Admin	3	3	2	3	17

Account Manipulation. Changing the password of an account does not require elevated privileges and does not change the permissions of the account. For persistence, we encounter an unusual case, since the owner of the account would be locked out of their account, thereby making it impossible to complete *User Action* while further maintaining the persistence of the adversary. Nonetheless, we will rate the persistence with 0 points because the owner of the account would likely initiate countermeasures such as an account deactivation or password reset, effectively undoing the adversaries' persistence. Password changes are logged through entries with event ID 4723, thereby being detected by monitoring. The reliability depends on the OS the same way it was in *Valid Accounts*. The full score can be seen in Table 6.

Table 6: Results of Account Manipulation.

	PS	PM	DE	RE	Score
Manipulation	0	2	2	3	7

Account Creation. Creating an account is only successful with admin rights. Nonetheless, these also enable us to elevate the privileges of the new account, thereby achieving effective administrator permissions. As in *Valid Accounts* the persistence and reliability are based on the basic functionality of an account, therefore achieving the same results. Account creation can be detected through the log entries with the event ID 4720. The full score can be seen in Table 7.

Table 7: Results of Account Creation.

	PS	PM	DE	RE	Score
Creation	3	2	2	3	16

Scheduled Task. We are required to have administrator rights in order to create a scheduled task. However, the tasks can be executed with elevated privileges. A task will maintain its persistence up to and including updates but will lose its persistence after a re-installation. All scheduled tasks show up in the Task Scheduler and can, consequently, be detected through monitoring. The task maintained its persistence with 100% reliability. The full score can be seen in Table 8.

Table 8: Results of Scheduled Task.

	PS	PM	DE	RE	Score
Task	3	2	2	3	16

Windows Services. Installing a new service requires administrator privileges and the service is executed using the local system account, thereby maintaining its permissions. Persistence is maintained up to and including updates, a re-install will also remove services installed by the user. All installed services can be seen in the *Services* application and are, therefore, detectable by monitoring. The service maintained its persistence successfully throughout all ten reboots. Table 9 provides an overview of the score.

Table 9: Results of Windows Services.

	PS	PM	DE	RE	Score
Services	3	2	2	3	16

Shortcut Modification. There are two different start-up folders into which we can place our shortcut, the current user and the common startup folder. In consequence, we will be differentiating between these two scenarios. We are able to place a shortcut in the current user startup folder with user privileges and into the common folder with administrator rights. Both scenarios are unaffected by user action, reboots, and updates. However, a reinstall will cause the contents of both startup folders to be deleted. Furthermore, both scenarios are detectable through monitoring because the shortcuts are visible within the folder. Both scenarios achieved 100% reliability. The full score can be seen in Table 10.

Table 10: Results of Shortcut Modification.

Shortcut	PS	PM	DE	RE	Score
Current User	3	2	2	3	16
Common	3	2	2	3	16

Registry Run Keys. As with *Shortcut Modification*, there are two different registry run keys, one specific to the current user and one common key. In consequence, we will again be considering two scenarios. Similar to shortcut modification, the current user key can be modified using user privileges and the common key requires admin rights. Persistence is maintained through user actions, reboots, and updates. Nonetheless, a re-install will cause a loss of persistence. Both scenarios are detectable through the Registry Editor and the execution of entries to both keys was 100% reliable. An overview of the scores for both scenarios can be seen in Table 11.

Table 11: Results of Registry Run Keys.

Key	PS	PM	DE	RE	Score
HKCU	3	2	2	3	16
HKLM	3	2	2	3	16

5.3 Windows Defensive Software

On each OS, the defensive mechanisms are set up in the same way. Without needing to start any scan, Windows Defender identifies `bind.exe` as malware across all four systems. Windows Defender flags the file as malware and prevents nearly all interaction (moving, copying, executing) with it. Shortcuts of the file can be created. However, their functionality is limited because the file cannot be executed. Furthermore, the file was often automatically deleted from the OS after a short time frame. This reaction was caused, as no defense evasion was used such as obfuscation. In addition, MSFvenom is commonly used by adversaries and is hence known by Windows Defender. The methods valid account, account manipulation, and account creation remain unaffected by Windows Defender since they do not use `bind.exe`. All other methods (task, service, shortcut, registry) were unable to establish persistence because they rely on the use of `bind.exe`.

6 RESULTS

When comparing each method's performance on different Windows OSs, no differences can be found. This does not come as a surprise, as all methods are core features of the Windows OS. However, a significant difference is found when comparing the performance of each method on an unprotected versus protected version of the OS. All methods, except for the three based on abusing accounts, were fully disabled, since they were not capable of establishing any persistence. This shows that Windows Defender is not capable of discerning normal account use and the abuse of an account.

Most methods performed similarly. Out of ten, eight scored equally in all aspects (16 points). Account manipulation performed significantly worse with seven points because even though it technically maintains persistence, it can practically be removed. Valid accounts for administrators performed slightly better with 17 points since it enables privilege escalation.

The similarities in each method's persistence and reliability likely stem from the fact that each is a system feature specifically designed to persist on an actively used system. Hence, they should function in a reliable way as important OS processes are often executed through these. Since privilege escalation is not a goal of persistence, it was to be anticipated that only a few methods would achieve this. This may explain why we see no method that scores at least one point in permission. The detection results were also to be expected because we did not try to actively hide our actions. More advanced malware will try to hide its activities, in order to stay undetected from Windows Defender among others.

In consequence, the metric provides a simple measure to compare different OS versions and methods used by adversaries. It can be applied to more advanced methods used by APTs. The results may guide future improvements in detection methods by defensive mechanisms.

7 CONCLUSION AND OUTLOOK

The ability to assess the effectiveness of methods compared to each other enables the understanding of the threat each method poses to IT infrastructure. Therefore, based on related work, we explained common persistence methods for Windows OS. Then we described our methodology with the workflow and metric for evaluating persistence methods. This methodology was applied on Windows 10, Windows 11, Windows Server 2019, and Windows Server 2022 using a bind-shell executable as exemplary malware. The evaluation was performed without and with defensive mechanisms activated. The results of the evaluation conclude that most methods performed very similarly, likely due to the fact that each mechanism abused a feature integral to the Microsoft Windows OS. The majority of methods are capable of reliably establishing persistence which could withstand updates of the OS. Enabling Windows' built-in defensive mechanisms resulted in a significant decline in the performance of persistence methods. This points to the effectiveness of current protection measures against commonly used persistence methods.

In the next step, more complex and real-world malware – with obfuscation methods – will be applied and analyzed for the distribution of applied methods. Last but not least, we plan to evaluate more as well as combinations of methods.

REFERENCES

Barr-Smith, F., Ugarte-Pedrero, X., Graziano, M., Spolaor, R., and Martinovic, I. Survivalism: Systematic Analysis of Windows Malware Living-Off-The-Land. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1557–1574. IEEE.

CISA (2022). Alert (AA22-074A). https://www.cisa.gov/uscert/ncas/alerts/aa22-074a. Accessed 17.11.2022.

Dubey, L. (2019). Identifying The Malware Persistence using Advance Static And Advance Dynamic Method. *International Journal of Scientific & Technology Research*, 8.

Gittins, Z. and Soltys, M. (2020). Malware Persistence Mechanisms. In *24th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems*, pages 88–97. Elsevier.

Hunkeler, A. (2022). Collection of malware persistence information. https://github.com/Karneades/malware-persistence. Accessed 17.11.2022.

Laurenza, G., Lazzeretti, R., and Mazzotti, L. (2020). Malware Triage for Early Identification of Advanced Persistent Threat Activities. *Digital Threats*, 1(3).

Mankin, J. L. (2013). *Classification of malware persistence mechanisms using low-artifact disk instrumentation*. PhD thesis, Northeastern University Boston, MA.

Microsoft (2021). Task Scheduler for developers. https://learn.microsoft.com/en-us/windows/win32/taskschd/task-scheduler-start-page. Accessed 17.11.2022.

MITRE (2022). MITRE ATT&ACK. https://attack.mitre.org. Accessed 17.11.2022.

Park, D. and Yener, B. (2020). A Survey on Practical Adversarial Examples for Malware Classifiers. In *Reversing and Offensive-Oriented Trends Symposium (ROOTS)*, pages 23–35, New York, NY, USA. Association for Computing Machinery.

Potter, S. and Nieh, J. (2005). Reducing Downtime Due to System Maintenance and Upgrades. In *19th Large Installation System Administration Conference (LISA 05)*. USENIX Association.

Rana, M. U., Ali Shah, M., and Ellahi, O. (2021). Malware Persistence and Obfuscation: An Analysis on Concealed Strategies. In *2021 26th International Conference on Automation and Computing (ICAC)*, number Portsmouth, United Kingdom, pages 1–6. IEEE.

Reischaga, Lim, C., and Kotualubun, Y. S. (2020). Uncovering Malware Traits Using Hybrid Analysis. In *Proceedings of the International Conference on Engineering and Information Technology for Sustainable Industry (ICONETSI)*, New York, NY, USA. Association for Computing Machinery.

StatCounter (2022). Desktop Operating System Market Share Worldwide – Sept 2021 - Sept 2022. https://gs.statcounter.com/os-market-share/desktop/worldwide. Accessed 17.11.2022.

Thomas, K., Li, F., Zand, A., Barrett, J., Ranieri, J., Invernizzi, L., Markov, Y., Comanescu, O., Eranti, V., Moscicki, A., Margolis, D., Paxson, V., and Bursztein, E. (2017). Data Breaches, Phishing, or Malware? Understanding the Risks of Stolen Credentials. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1421–1434. Association for Computing Machinery.