Computers & Security 121 (2022) 102855

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

A Taxonomy for Threat Actors' Persistence Techniques

Antonio Villalón-Huerta[a,*], Hector Marco-Gisbert[b], Ismael Ripoll-Ripoll[b]

[a] S2 Grupo, Ramiro de Maeztu 7, Valencia, 46022 Spain
[b] Department of Computing Engineering, Universitat Politécnica de Valéncia, Camino de Vera s/n, Valencia,46022 Spain

ARTICLE INFO

Article history:
Received 18 March 2022
Revised 22 June 2022
Accepted 21 July 2022
Available online 26 July 2022

Keywords:
TTP
Persistence
Advanced Persistent Threat
Malware
MITRE ATT&CK

ABSTRACT

The main contribution of this paper is to provide an accurate taxonomy for Persistence techniques, which allows the detection of novel techniques and the identification of appropriate countermeasures. Persistence is a key tactic for advanced offensive cyber operations. The techniques that achieve persistence have been largely analyzed in particular environments, but there is no suitable platform–agnostic model to structure persistence techniques. This lack causes a serious problem in the modeling of activities of advanced threat actors, hindering both their detection and the implementation of countermeasures against their activities. In this paper we analyze previous work in this field and propose a novel taxonomy for persistence techniques based on persistence points, a key concept we introduce in our work as the basis for the proposed taxonomy. Our work will help analysts to identify, classify and detect compromises, significantly reducing the amount of effort needed for these tasks. It follows a logical structure that can be easy to expand and adapt, and it can be directly used in commonly accepted industry standards such as MITRE ATT&CK.

1. Introduction

Persistence refers to the capability of a malware to survive to changes and interrupts, including reboots, in a compromised system. It is a widely used term in malware research, so most of its definition are malware–related; for example, Sikorski and Honig (2012) define persistence as a behaviour of malware by which it tries to be in a compromised system for a long time. A more accurate definition is provided by Gittins and Soltys (2020), where the authors define persistence as a method by which malware survives a reboot of the victim operating system. Persistence is an important attribute that malware writers consider in its design; the reason is simple (Wei et al., 2017): the longer the malware can stay in the victims device, the more value it generates for the adversary.

However, persistence is not malware–specific: advanced threat actors do not use malware in all operations, but they also try to maintain their persistence in a targeted victim. In fact, persistence is a key tactic for these actors; when an Advanced Persistent Threat (APT) compromises a victim, one of the first steps that it will execute is to guarantee its foothold on the targeted infrastructure, maintaining the compromise upon system reboots.

Considering persistence as a key goal, persistence techniques are those that allow a threat actor to achieve it; they have been largely analyzed, but we have not found a clear structure for them, being most of the approaches architecture and operating system (OS) dependent. This fact hinders the identification of compromises out of the analyzed technologies. This paper provides a platform–agnostic taxonomy for persistence techniques, suitable for their identification and characterization. This taxonomy is based on a concept we introduce in our work, the persistence point: the location within the system where a persistence artifact is stored.

The contributions of this paper are the following ones:

- To provide an OS–independent taxonomy of persistence points, thus allowing analysts to identify persistence techniques in all platforms.
- To ease not only the detection of persistent artifacts but also the identification of the capabilities of threat actors.
- To identify the most used persistence points, then determining the most exploited system capabilities to achieve persistence and establishing probabilities in order to prioritize investigations, where applicable.

The rest of the paper is organized as follows. The background, Section 2, provides an introduction to cyber operations tactics and techniques. In Section 3 we assess the problem of the lack of a suitable structure for persistence techniques to help analysts in

* Corresponding author.
E-mail addresses: antonio.villalon@s2grupo.es (A. Villalón-Huerta), iripoll@disca.upv.es (H. Marco-Gisbert).

https://doi.org/10.1016/j.cose.2022.102855

their investigations. Section 4 analyzes the prior work to identify persistence approaches, and in Section 5 we propose a novel taxonomy for persistence points, as a direct way to structure techniques and to identify a compromised system. In Section 6 we discuss the results of our work, comparing them with previous approaches and identifying improvements where applicable, as well as future research lines. Finally, Section 7 summarizes the outcome of the overall work.

2. Background

From an abstract point of view, Robert Axelrod et al. define persistence of a resource (Axelrod and Iliev, 2014) as the probability that if you refrain from using it now, it will still be usable in the next time period. When specifically dealing with malware, persistence is defined (Kirillov et al., 2011) as a process by which malware ensures continual execution on a system, independent of low-level system events such as shutdowns and reboots.

Persistent malware is the one (Vogl et al., 2014) that makes permanent changes in memory and permanently changes the control flow within a system such that it can continue to achieve its objective. This feature allows the malware to be aware of and to react to changes in the compromised systems. Without persistence, malware is severely limited, and consequently, its impact is also limited; for this reason most threat actors, particularly advanced ones, will establish persistence in their victims in order to achieve an extended temporal effect for their operations.

In the context of advanced threat actors' operations, persistence is the tactic by which adversaries try to maintain their foothold on a targeted infrastructure; this tactic comprises different techniques that include any access, action, or configuration changes that allow the attacker to achieve the tactic, such as replacing or hijacking legitimate code or adding startup code. As we can see, this concept of persistence as a tactic exceeds all technological –including malware– considerations: persistence is defined as a key tactic for threat actors, regardless of malware. To accomplish its goals, an advanced threat actor must guarantee its presence in a compromised infrastructure: that is, in one or more compromised systems.

3. Problem statement

Persistence is a key tactic for advanced threat actors and even for simple malware: the ability to control a compromised target over time, while remaining unnoticed, is an essential to guarantee the success of an offensive operation. However, when analyzing the capabilities that these threat actors develop in their operations, all research is focused on particular approaches related to specific technologies. In this sense, Microsoft Windows environments, as the most abused platform, have been largely analyzed in order to identify the locations where the artifacts that grant persistence can be located.

An analysis focused on specific platforms, or even on specific malware, has obvious limitations, as all the identification of persistence techniques are architecture and OS–dependent. The most relevant limitation is related to the identification of techniques in technologies outside of the scope of those specific platforms. This means that when faced with the compromise of a new platform and trying to identify if an attacker has enabled persistence on it, an analyst has only vague references for investigation. In this case, analysts do not have a common reference to identify the mechanisms that enable persistence, nor to identify the locations of the system where an intruder can achieve persistence. For an agile identification of persistence techniques in new environments, analysts need a common platform–agnostic reference that allows them to quickly determine which elements have to be examined and what to look for inside them.

In addition to the dependence on specific platforms, much research work about persistence focuses on specific malware and its capabilities to survive a system reboot. These approaches are useful for the analysis of particular malicious code, but they lack a global vision of the persistence. As we have stated before, persistence is a key tactic for an advanced threat, not a simple malware capability. In fact, different APT actors do not rely on malware to conduct their operations, the so called malwareless ones, while others do not use malware to achieve some specific tactics. Please note that although in computer virology the definition of malware is an open problem (Kramer and Bradfield, 2010), when we refer to malware we refer to any code added, changed, or removed from a software system in order to intentionally cause harm or subvert the intended function of the system (McGraw and Morrisett, 2000; Namanya et al., 2018; Mishra and Jha, 2022). Attending to this definition, we do not consider elements such as legitimate system tools or remote credentials for legitimate services as malware, although they can be abused to maintain persistence.

When facing malwareless operations, detection becomes more complex. Atomic and computed indicators of compromise, such as hashes, do not provide the full capabilities for an accurate detection, so analysts must usually deal with behavioral indicators: for example, those that allow the contextualization of a legitimate system tool execution in order to classify it as anomalous. Due to their stealthiness, malware free approaches are interesting for advanced threat actors in different tactics of an operation, from intrusion (Zimba and Wang, 2017) to data exfiltration (Zimba and Chishimba, 2017) or lateral movement (Ussath et al., 2016). Regarding persistence, different techniques exploited by threat actors such as SANDWORM (Slowik, 2018) or APT29 (Nafisi and Lelli, 2021) do not rely on malware but on legitimate services and tools. Even some actors such as ALLANITE are able to conduct whole malwareless operations (Slowik, 2019). In this way, we argue that persistence is not only malware–related, but it is a tactic of advanced threat actors in offensive cyber operations that can be achieved through malware or through simple abuse of legitimate resources.

No suitable approximation for persistence techniques that would allow analysts the identification of compromised systems has been defined. In addition, no research has been performed regarding where persistence is stored in a targeted system: that is, which locations of a system should be analyzed in order to detect the compromise. Identifying where persistence is stored is a must when dealing with operations performed by advanced threat actors. Tactics are mandatory to identify what to look for, but detecting where the attacker has stored an artifact for persistence is mandatory to identify where to look for techniques that implement the persistence tactic. Without a common platform–agnostic reference for persistence techniques, the detection of persistence artifacts in a targeted system follows an unstructured approach that delays the defensive capabilities and opens a window of opportunity for hostile actors.

4. Approaches and limitations

MITRE ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge) is a globally accessible knowledge base of adversary tactics and techniques based on real–world observations. This framework represents the biggest effort to identify tactics and techniques for advanced actors, and it presents a plain structure for Persistence techniques, which makes it difficult to establish a clear taxonomy or categorization for them. Some efforts have been made to align technical persistence capabilities with MITRE ATT&CK. Oosthoek and Doerr (2019) provides a link between Microsoft Windows malware and the framework by identifying the malware persistence capabilities. These relationships have also been ana-

lyzed in Gittins and Soltys (2020) and Lee et al. (2020), where the authors map different persistence approaches used in Microsoft Windows malware samples to MITRE ATT&CK techniques. However, in the case of the Persistence tactic, this framework is too close to the most technical aspects of techniques, so it is not suitable for abstraction and for a modeling of techniques in a platform–agnostic way. Other malware–related frameworks from MITRE, such as MAEC (Malware Attribute Enumeration and Characterization) or CME (Common Malware Enumeration) do not provide even a categorization for persistence techniques. SHIELD, an active defense knowledge base that MITRE is developing to capture and organize what they are learning about active defense and adversary engagement, identifies defenses against the techniques stated in ATT&CK, but it does not provide a valid taxonomy for them.

Sharma et al. (2019) characterize malware persistence into three categories: run–time, re–boot and trojanized system binaries; although this can be considered an interesting approach to consider behaviors of the malware in order to design a Fuzzy Inference System, it does not capture all the elements to build a taxonomy model.

Based on their basic approach to persistence, Webb (2018) classifies techniques in five main categories: User Login Execution, System Startup Execution, Dynamic Linked Library (DLL) Injection, Execution Hijacking and Adversary Backdoors. This proposal provides some key aspects to a OS–independent approach, but as it is mainly focused on Microsoft Windows aspects, it still lacks a complete platform–agnostic view.

Mankin (2013) breaks persistence capabilities into three phases: installation, system boot and service load. Although this is an interesting approach for the modeling and detection of persistence, the author does not propose a classification for persistence mechanisms, but a common breakdown structure for all of them, with a focus on Windows services as a particular technique.

Most of the current approaches to the analysis of persistence techniques have been made to identify OS–dependent persistence methods. As we have stated before, the most analyzed persistence mechanisms are related to Microsoft Windows environments. Windows Auto–Start Extensibility Points (ASEP) were first introduced by Wang et al. (2004); they were defined as the subset of operating system and application extensibility points that can be “hooked” to enable auto–starting of programs without explicit user invocation. ASEP are a key concept for malware persistence, as they define points that an attacker can abuse to maintain its foothold on a targeted system. In Uroz and Rodríguez (2019) Daniel Uroz et al. propose a taxonomy for Windows ASEP divided into four categories: system persistence mechanisms, program loader abuse, application abuse, and system behavior abuse; each of them is analyzed and its characteristics are extracted, identifying families of persistence points as shown in Fig. 1. Although this is a valid classification, in addition to being focused on Microsoft Windows it only defines the main four previous categories, without sub categories for them, so it should be detailed in order to specify a more complete taxonomy. A key resource to identify persistence points in Windows environments is Microsoft Autoruns (Russinovich et al., 2009; Russinovich and Margosis, 2016). It is a utility that has the most comprehensive knowledge of auto–starting locations of any startup monitor, showing what programs are configured to run during system boot up or login, and when various built–in Windows applications are started.

The technical persistence capabilities that Microsoft Windows provides have been analyzed in different works. Monnappa (2018); Sikorski and Honig (2012) and O'Leary (2019) identify particular persistence mechanisms and locations for this operating system, while (Mohanta and Saldanha, 2020) provides a basic structure for persistence capabilities: startup shell directories, registry RUN, services, file infection, DLL hijacking, Winlogon and Task Scheduler. These approaches provide different proposals for the identification of persistence techniques in Microsoft Windows. However, none of these approaches, all of them focused on a particular environment, provides a platform–agnostic proposal suitable to be used in other technologies.

Regarding other operating systems, in Hwang and Lee (2019) Jun–ho Hwang et al. identify four methods for ELF malware persistence in Linux systems: subsystems initialization, time–based execution, file infection and replacement and user files alteration. O'Leary (2019) analyzes both Windows and Linux persistence mechanisms with practical examples, but without establishing a common framework for the classification of the identified techniques. Wardle (2014b) analyzes malware persistence mechanisms in Mac OS X, as well as the particular techniques used by different malware samples in this operating system. Also regarding Mac OS X, Wardle (2014b) provides an initial approach to technical capabilities and analyzes several malware samples and their persistence techniques. Again, as in Windows environments, none of these approaches delves into the definition of a general proposal for persistence techniques, considering only the identification of particular techniques for specific platforms.

Another key research line for persistence techniques is IoT–focused malware, as connected devices are growing in number and critical infrastructures are a clear target for hostile actors. Linux IoT malware persistence capabilities are analyzed in Brierley et al. (2020), where Calvin Brierley et al. identify, without providing a model or structure, six methods for persistence: modifying writable file systems, recreating read–only file systems, initrd/initramfs modification, “set writable flag” kernel module, update process exploitation and ubootkit. The authors defend that no universal method to gain persistence on IoT devices has been identified. In Bytes and Zhou (2020) Andrei Bytes et al. analyze techniques used in Programmable Logic Controllers (PLC), extending Linux generic mechanisms to particular embedded Linux devices but without providing a suitable structure for them. Németh (2020) analyzes rootkit persistence techniques in IoT devices, identifying Linux Kernel Modules, ramdisk–based and user space programs as main categories. Inside the last of them, the authors classify techniques into the following classes: service managers, job schedulers and user–specific startup files. Related to smart grid environments, in Eder-Neuhauser et al. (2017) Peter Eder–Neuhauser et al. identify persistence methods such as manipulation or anti malware tools, code obfuscation or encryption techniques. The authors analyze malware samples using each of the identified persistence mechanisms, but they do not provide a general structured classification suitable for its use outside smart grid environments.

In addition to different operating systems technologies, the persistence capabilities of malware families are also a key research focus. One of the most analyzed families is ransomware, due to the growing impact that this malicious software is causing in all kind of organizations. Lemmou et al. (2021) different ransomware samples and their persistence mechanisms, among other behaviors. Regarding banking malware, which is also a relevant issue, Black et al. (2018) an identification of the mechanisms for persistence in this kind of malicious programs: specific registry keys, such as registry or AppInit_DLLs, program trojanization, and bootkits.

Persistence capabilities from specific malware samples have been also analyzed in different works. Gittins and Soltys (2020) analyzes several samples to identify their persistence capabilities. In Popli and Girdhar (2019) the authors analyze WannaCry and Petya capabilities without focusing on an in–depth analysis, identifying the persistence capabilities of both malware samples.

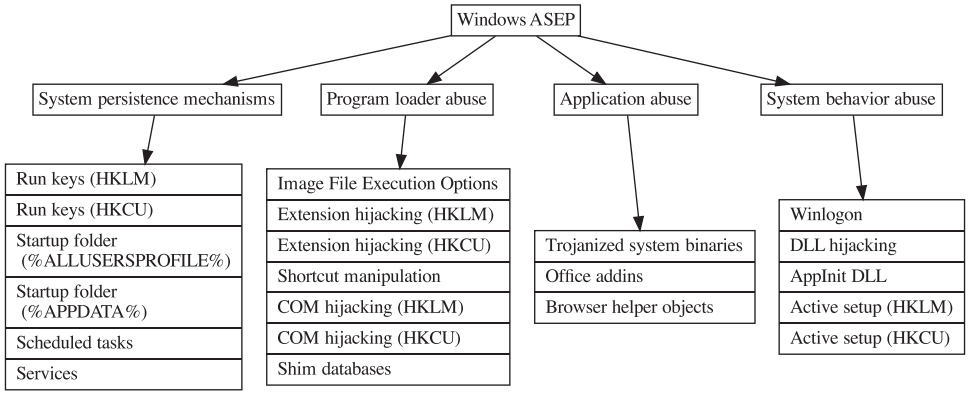

Fig. 1. Windows Auto–Start Extensibility Points

Akbanov et al. (2019) and Kao and Hsiao (2018) provide an in–depth analysis of WannaCry, and Wardle (2014a) analyzes iWorm persistence, an OS X backdoor. Although these works analyze the persistence techniques of main malware samples or families, none of them provides a suitable structure for the abstraction of these techniques.

Malware that does not rely on the file system to run is referred to as fileless malware, and its persistence capabilities have been analyzed in different works. In Kumar et al. (2020) Sushil Kumar et al. classify fileless malware in three different categories in relation to its persistence techniques: memory–resident malware, Windows registry malware and rootkit fileless malware. In Sanjay et al. (2018) the authors identify Windows registry, WMI store, SQL tables or Scheduled tasks as usual locations to achieve fileless persistence. In Vogl et al. (2014) Sebastian Vogl et al. analyze persistent data–only malware and discuss its challenges, presenting a proof of concept of this kind of malware and providing additional techniques to achieve persistence. Data only malware is malware that introduces specially crafted data into a system with the intent of manipulating the control flow without changing or introducing new code. Ramaswamy (2008) provides an initial classification for rootkits, differentiating between kernel–level and user–level rootkits, being the first ones the most analyzed in literature. In Joy et al. (2011) Jestin Joy et al. expand this classification, identifying three types of rootkits: virtualization, kernel–level and library–level, being this last one a user–level rootkit. Nevertheless, none of the related work on malware persistence provides a taxonomy for the persistence techniques they explore, focusing once again only on the particular features of samples, families or malware persistence approaches.

5. Our proposal

After the analysis of the current approaches to define a valid classification for persistence techniques, their strengths and weaknesses and, especially, their main lacks, we propose a platform–independent persistence taxonomy. This taxonomy will allow analysts to classify techniques, regardless of the technology used in each case, as well as to identify new persistence capabilities that threat actors might develop, thus being able to identify and implement counter measures against them.

To establish such a taxonomy we have to formally define the following concepts:

- Persistence point. The location within a compromised system where a persistence artifact is stored.
- Persistence technique. The actions that enable a persistence mechanism into a target, relying on a persistence point.

Please note that in this context, the term “artifact” refers not only to malicious software implanted in the targeted system, but to any software or configuration abused or manipulated by a threat actor in order to gain persistence. A persistence point is, as its definition highlights, a location; this location can be a hardware component, a file system point, a Windows registry entry, etc. As an example, in previous sections we have referred to fileless malware; regarding persistence capabilities, this could be considered a first classification for persistence points, although it is too general, as it comprises many types of persistence points with little relation between them.

Persistence techniques directly rely on persistence points to achieve their goal; from an analysts’ perspective, a persistence point defines where to look for persistence, while a persistence technique defines what to look for. No technique can be achieved without a persistence point. In this context, we can understand a persistence technique just as the abuse or manipulation of a persistence point. We refer to abuse when a threat actor does not modify the persistence point, but just exploits one or more of its standard capabilities. We refer to manipulation when the attacker alters the persistence point for his own benefit, by fabricating spurious data or by modifying or canceling legitimate data. For example, techniques relying on system accounts as a persistence point include those related to credential abuse, those related to the modification of legitimate users’ credentials and those related to the addition of non legitimate users to the system. The relationship between persistence techniques and persistence points is direct: all persistence techniques rely on at least one persistence point, and we can detect the artifacts that achieve persistence by inspecting those points.

In this work we establish a taxonomy for persistence points, which directly defines a taxonomy for persistence techniques. We propose four upper level categories:

- Pre–OS persistence points, those regarding hardware, firmware or initial sequences of a system boot, before a particular operating system is loaded.
- OS persistence points, those related to the boot of a particular operating system and to its native capabilities.
- Server–software persistence points, those related to remotely accessible software that is provided to users without a full access to the system.
- User persistence points, those related to particular user activities or configurations.

In the following sections we discuss each of the proposed categories for persistence points and we specify their particular structure, thus defining a taxonomy for persistence techniques.

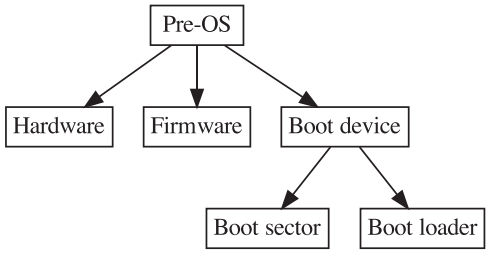

Fig. 2. Pre–OS persistence points

5.1. Pre–OS persistence

The boot process for a computer system fully depends on the specific hardware and on the OS that will be running on it. Most research is focused on the boot process for BIOS or UEFI based x86 systems running Microsoft Windows OS flavours. In this particular case, the boot process can be described in a simple way as follows:

1. On startup, firmware (BIOS or UEFI) is called and transferred with execution.
2. This firmware initializes hardware devices and goes through storage devices to look for a bootable one.
3. When a bootable device is found, boot code is executed, loading the boot sector for a bootable partition.
4. Boot sector loads and executes the operating system boot loader.
5. Boot loader loads the OS kernel from the storage device.

In the case of UEFI systems, this firmware directly loads the operating system boot loader without relying on boot sectors, thus skipping step 3; in these systems, this boot loader is just an EFI file in the filesystem.

While a rootkit is malicious software that impacts its target at user and kernel levels (Rao and Selvakumar, 2014), a bootkit is a particular rootkit that transfers its storage location from the file system to the hardware and activates itself while or even before the operating system kernel is loaded (Li et al., 2011). Bootkits are largely analyzed in Matrosov et al. (2019); as this malicious software is loaded before the operating system, it can tamper the whole computer system. According to the different stages in boot process, Li et al. (2011) classifies bootkit technologies into four categories: BIOS–based bootkit, MBR–based bootkit, NTLDRbased bootkit, and others. A similar approach is followed in Gao et al. (2012), where Hongbo Gao et al. expose the same classification without the “others” category. As BIOS–based bootkits write their malicious payload directly into the BIOS, they typically have to target particular BIOS or hardware, so they are rarely seen in the wild (Grill, 2016). MBR and NTLDR–based bootkits are the dominant types in real–world malware.

These works provide a key approach for an initial taxonomy of Pre–OS persistence points: those that are OS–independent, BIOS or MBR, and those that are based on the initial steps of a particular OS being loaded, such as NTLDR and others. However, this approach lacks a whole family of persistence points that we must consider: those related to hardware implants. In fact, hardware implants are not usually considered in persistence techniques families, such as those presented in MITRE ATT&CK.

To provide a platform–agnostic approach, we divide Pre–OS persistence points into three main families: hardware, firmware and software related, as shown in Fig. 2.

Hardware persistence points are those related to hardware implants to maintain persistence on a specific targeted system. At the early stages of a system boot, this hardware and its linked firmware are initialized by a firmware component, just as UEFI or BIOS. These implants have few academic research, especially focusing on covert channels for air–gapped networks (Guri et al., 2016; Wakabayashi et al., 2017; Morgner et al., 2018). In LaSota (2019) Austin Lasota provides a very brief introduction to different types of hardware implants in Apple’s Mac hardware and their countermeasures. Until now, the most extensive information about these implants continues to be NSA’s ANT catalog, which has been publicly exposed and which is detailed in works such as Cayford et al. (2014).

In addition to hardware, UEFI, BIOS or equivalent firmware is another key persistence point. In Matrosov (2019) Alex Matrosov proposes a classification of vulnerabilities and attack vectors for BIOS persistent infection, identifying persistent and non–persistent implants for UEFI firmware through post exploitation and supply chain vulnerabilities. Note that when referring to hardware and firmware persistence in our taxonomy we are considering not only the main system components, but also the peripheral devices, especially those that have direct memory access (DMA) to the main system runtime memory. In fact, DMA attacks have been largely analyzed, from a pure malware perspective in Stewin and Bystrov (2012), where Patrick Stewin et al. introduce the concept of DMA malware, to a general threat actor capability (Breuk and Spruyt, 2012).

Finally, firmware components go through a boot sequence that depends on the type of firmware being used; here we find the third family of Pre–OS persistence points, those related to the boot device. In this case we differentiate two families of persistence points: boot sector and boot loader ones. Legacy systems (i.e., BIOS) use boot sectors (for example, Master Boot Record, or MBR, which loads a Volume Boot Record, or VBR). Boot sectors are to call a boot loader (in those systems with UEFI this boot loader is just an EFI file in the filesystem, as stated before), which loads the OS kernel, which is another persistence point but in this case OS–dependent, as described in next section.

We want to highlight that when dealing with virtual machines running over hypervisors, our proposed taxonomy is completely valid. In this particular case, pre–OS persistence points are ones exposed in this section, but considering two main aspects. The first one is that these persistence points can be identified in the hypervisor systems, not only in the virtual machines. The second one regards these virtual machines: being the same persistence points, on virtual machines they are stored in software that emulates hardware components.

5.2. OS native persistence

Once the OS starts to boot, a second category for our taxonomy is defined by those persistence techniques based on different OS capabilities. In this case, the obtained privileges can be those of the operating system or those of a particular user, as system boot is executed with administrative privileges. In Fig. 3 our proposed OS native persistence points taxonomy is shown.

Within this category we find first of all the operating system kernel as a persistence point; when an OS boots, its kernel is loaded and a process is started to execute tasks such as software initialization or module loading. By subverting the operating system kernel, a kernel rootkit embeds itself into the compromised kernel and stealthily causes damage with full unrestricted access to the systems resources (Riley et al., 2008). Kernel persistence can be achieved through the compromise of the OS kernel itself or through the compromise of kernel modules in any of their forms, such as kernel modules, kernel extensions or kernel drivers, loaded on startup or when certain conditions are met. Malware such as Drovorub exploits kernel persistence points, establishing persistence through kernel modules (FBI/NSA, 2020).

Once the kernel is loaded and the basic capabilities of the operating system are started, boot procedures are another OS–

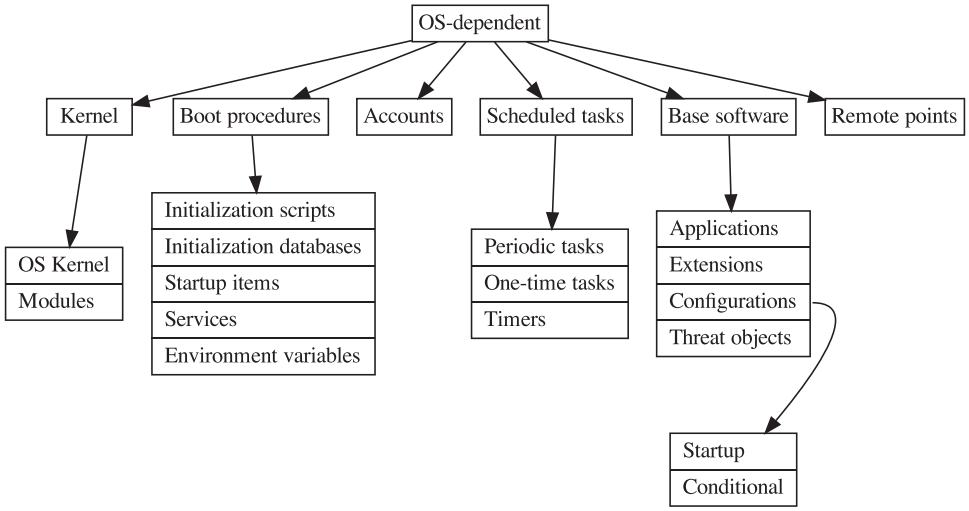

Fig. 3. OS native persistence points

dependent persistence point. In this stage of a system boot we find clock synchronization, daemons launched or subsystems initialized; in addition, please note that many operating systems can execute or load user files when booting. In all of these tasks, code is executed and configuration files are loaded, so a hostile actor can modify them to achieve OS–dependent persistence; in the particular case of remote objects access, such as in clock synchronization, the threat actor can also compromise remote systems in order to establish persistence on a targeted one, but here we must refer to remote persistence points, as stated in this section.

Inside the boot procedures family, in addition to these initialization scripts, many operating systems provide a database, such as Windows Registry or AIX Object Data Manager, where stored data is accessed during system boot to initialize specific OS capabilities, such as network settings or program launch. We must differentiate this family of persistence points from the previous one, as they are stored on a different location for the OS and they are accessed and managed in most cases with specific native tools. In fact, malware that enables persistence by adding a specific entry into these databases is considered fileless, as opposed to malware that stores an artifact in the filesystem. For example, Windows Registry is a widely used persistence point; we can find different threat actors exploiting this registry to achieve persistence, such as APT32 (Dahan, 2017) or APT37 (FireEye, 2018), as well as specific malware such as WannaCry (Akbanov et al., 2019) or Bisonal (Hayashi and Ray, 2018; Horejsi et al., 2020).

In addition to initialization databases, different operating systems such as Microsoft Windows or Apple Mac OSX also provide a startup items location, usually a folder with references to programs (in many cases in the form of symbolic links) that are executed during system boot. By simply adding the correct reference to a program in this persistence point, the program will be launched on startup; jRAT is a known cross–platform backdoor exploiting this kind of persistence points (Kamluk and Gostev, 2016). Please note that in these operating systems there is usually a similar persistence point for each particular user, where references are executed when the user logs in.

As a fourth category inside the boot procedures family, we find services, also called daemons, managed by the OS and launched to perform specific tasks in the background. Linux malware often exploits these services persistence points (Cozzi et al., 2018; O'Leary, 2019). The tasks launched by services may include starting clients to connect to remote services, enabling operating system capabilities such as the execution of scheduled tasks or the launching of long time running processes that will be up as the operating system is running. Inside this category we find standard, mandatory OS services and daemons, those related to non operating system native applications: although they can also be started as services in many cases, from a technical point of view, we classify their persistence points inside the "Server Software" category. For example, the compromise of a Microsoft Exchange Server in order to establish persistence by a threat actor relies on Microsoft Exchange (a server software) as a persistence point, not on a native system service. This point is clear, as server software has its own configurations, in many cases including access accounts, outside the OS configuration files. While a system service can not be stopped or uninstalled without introducing some kind of system instability, a server software can be completely stopped interfering only with its own availability.

Finally, as a fifth family inside this category we must consider environment variables; these variables are set during system boot and they affect global execution of applications in the system. We differentiate them from the base software categories, as in this case the persistence point is not related to a particular application, but to the whole system. In addition, please note that many of these variables can be overwritten by user–related ones; in this case we do not consider them as a separate persistence point, as they are set in the login scripts for a particular user.

One of the easiest points for an attacker to gain persistence is through the abuse of accounts that grant access to targeted systems. This technique has been exploited by threat groups such as APT29 (CISA, 2020). When dealing with persistence points, we must differentiate accounts related persistence points from login–related ones; while all of them trigger execution when a user logs in the system, account–related ones are stored in the system's user table, while those related to user's login are stored in the user's home directory. In the same way, we must also differentiate system accounts, those that grant access to the system, both as privileged and as unprivileged users, and application accounts, those that grant access to a specific application served within the system, mainly to external users: for example, a web, database or e–mail account. An attacker can abuse both of them, but persistence points are different: in one case, the persistence point is located in the system's user table, while in the other one the persistence point is the particular user database or equivalent regarding the targeted application. For this reason we consider application accounts as a category for persistence points within software compromise, not within the accounts category.

The abuse of scheduled tasks capabilities that any OS provides comprises persistence techniques exploited in the wild by threat actors such as APT3 (Bahrami et al., 2019) or specific malware such as Emotet (Kuraku and Kalla, 2020), and even frameworks such as Cobalt Strike provide this capability (Varlioglu et al., 2022). Unix `at` or `cron` utilities, or Windows `at.exe` or Task Scheduler are commonly used by threat actors to maintain persistence on compromised systems. In this case, we can find three families of persistence points, as they are stored in different locations in the operating system: those related to the execution of periodic tasks, those related to the execution of one–time tasks and those related to timers.

Base software represents another family of OS–related persistence points in our taxonomy. In this case, persistence is stored in software that is not initialized during system boot (as it is in the "Services" branch) but in software that is natively provided by the OS, stored at disk and that is not mandatory for the OS to boot, but it is to run properly. In this category of persistence points we propose in our taxonomy four families: the own application and its extensions (that is, the binaries launched on the execution), the software configuration and threat objects.

The binaries used by the base software can be trojanized by a threat actor in order to execute malicious code, and this can be done both in the main software binary and in its extensions (for example, libraries, plugins or code loaded under certain circumstances). In this context, we define trojanization (Mankin, 2013) as the process to hijack an executable object that already exists on the system, patching it with malicious code that will be executed when the previously–benign program is loaded to run. An example of a threat actor achieving persistence through this persistence point is Gelsemium, a cyber espionage group that drops a trojanized DLL to be loaded by the spoolsv Windows service (Dupuy and Faou, 2021).

Software configuration is also a persistence point abused by attackers; in this case we refer to the configuration loaded on startup and to the configuration loaded under certain events (for example, when a condition is met or when a software extension is executed), named conditional configuration in this work. Please note that startup and conditional configuration persistence points are differentiated because they may be stored in different locations and they can also have different syntax and even different formats.

Finally, the last main category inside the base software persistence points is the one related to threat objects for the software. These malicious objects are especially crafted to execute certain actions on the targeted system when they are accessed in any form (for example, loaded or executed) by the software; until that moment, no malicious activity is performed, being this access the trigger that maintains persistence. Webshells are well–known examples of threat objects: malicious files added to the web contents that, when accessed in some form, can grant access to the system. Webshells are exploited by many threat actors such as Deep Panda (Thompson, 2020) or OilRig (Lai et al., 2021).

In our proposed taxonomy, we consider remote persistence points as the last family of locations that persistence techniques rely on. In this case, the persistence point is located on a remote system and persistence is triggered when the remote object is accessed. Although a remote persistence point can be found for almost all the previous categories, it is important to differentiate them as they are not stored on the targeted system, so this system has no hostile activity until the remote point is accessed. Remote persistence points can be found from the own boot process, for example in netbooted systems where the operating system boots from a network image, but also when dealing with accounts, remote services or even remote threat objects opened by a server software. We highlight that we consider important to differentiate them from local persistence points because they are stored outside the targeted system, so the techniques for their location and identification are different. Apart from laboratory proofs of concept, we have not identified specific real–world malware relying on remote persistence; although technically possible, we consider it is not exploited on the wild.

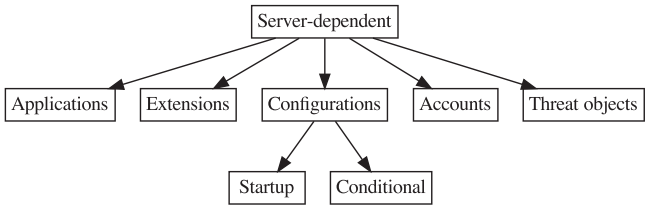

Fig. 4. Server–dependent persistence points

5.3. Server software persistence

Server software is another key persistence point. This category is related to software that, when running on a system, provides services to remote users; we must differentiate this family from the category of base software because server software is not mandatory for the operating system to run properly. With the exception of appliances or dedicated systems, server software is installed apart from the OS, so it can be uninstalled without affecting the OS native capabilities. Inside this category we can find software such as mail or web servers, VPN hubs or terminal servers.

In our taxonomy we propose five families of persistence points for server software persistence. We are identifying persistence points related to software, so four of them are the same as in base software, and also as the ones in user software: the own application and its extensions, the software configuration and the threat objects. The fifth of these families is related to the accounts that grant access to the software. In Fig. 4 this taxonomy is shown.

We must focus on this fifth category for server software persistence points, the one related to Accounts. In OS base software or in user software we do not find a family for Accounts, as in these cases software is executed in a system by a previously authenticated entity. However, when dealing with server software, Accounts are a key persistence point, as they are remotely abused by attackers. In server software, persistence techniques include the abuse of legitimate accounts as well as their manipulation. Account abuse relies on valid software credentials that are used by a threat actor among time, granting direct access to the software and to the information. Known threat actors abusing accounts to gain persistence are APT28 (Mwiki et al., 2019), APT29 (Gavaudan et al., 2021) or APT39 (Hawley et al., 2019). Account manipulation include the addition of accounts to be exploited by an attacker, as well as the modification of credentials that grant access to the software. Known threat actors performing account manipulation for persistence purposes include Sandworm (Slowik, 2018).

5.4. User dependent persistence

Finally, a fourth category for persistence points, as for techniques, is the one based on specific user locations and actions. In these techniques, persistence is triggered after a user executes a particular action, and the persistence point is usually located in the home directory of the targeted user, with exception of remote persistence points, as we will describe later. The privileges of the hostile actor are those of the particular user that executed the action, being the capabilities to execute privileged commands restricted to the elevation of privileges through the exploitation of vulnerabilities. In Fig. 5 our proposed User–dependent persistence points taxonomy is shown. Please note that although an attacker can enable persistence relying on the user's scheduled tasks, this family

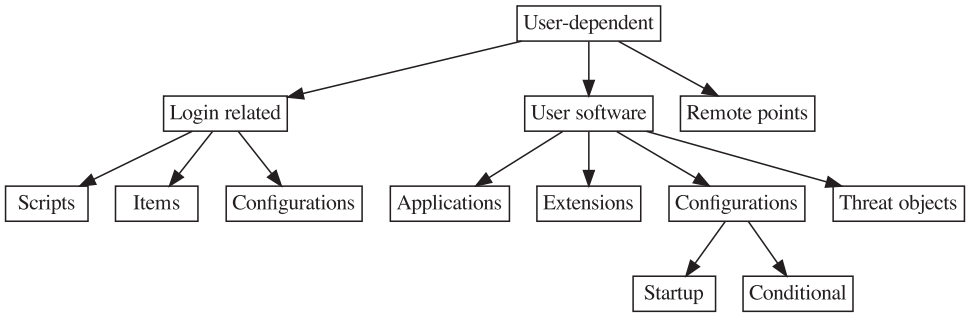

Fig. 5. User–dependent persistence points

of persistence points is considered OS–dependent, as it does not rely on a specific user action to be triggered, but on the complete boot of the OS.

In first place, login–related techniques are those that group persistence points triggered when a user logs in a system. We must consider this category apart from the one exposed in the previous section, regarding accounts of a targeted systems, as the persistence point in this case is different and located in the user's own configuration, not in the system capabilities.

Inside the login–related persistence points we must differentiate between logon scripts, logon items and logon configurations. The first family includes those user–defined files that are executed when the user logs in the system, usually in the form of scripts. The second one refers, as when dealing when OS–booting related persistence, to the location where references to applications are stored to be automatically launched when a user logs in. Finally, the third family refers to specific software configurations that are loaded in the login process, not by user software but by server software that enables the login process. For example, if a threat actor achieves persistence by modifying a user SSH "authorized_keys" file, it is not altering a user software configuration, but a user configuration loaded by a server software. These persistence points could be considered as conditional configurations for server software, but we find it important to differentiate them, as they are located in a different place from global server software configurations, and they are also writable without privileged access to the system. When we are referring to login related persistence points, we must also consider logout as a persistence trigger. Most operating systems allow users to define scripts, items or configurations to be accessed not only while a user logs in, but also when a user logs out of the targeted system. APT28 is a threat actor which actively exploits Login related persistence points in Windows systems (Calvet et al., 2016); examples of malware which is also able to exploit these points include Attor (Hromcová, 2019) or Netwire (Chen et al., 2020).

Another main family of user–persistence techniques is the one based on user software. When we refer to user software we are dealing with applications that are executed by specific users, not by system capabilities, although in most cases, particularly in multi–user environments, the software itself and some of its extensions are not writable by a normal user, but only by a privileged one. For user software as a persistence point we propose a structure similar to the one regarding server software, with the exception of the "Accounts" category, as user software does not use this kind of persistence point. Following this approach, persistence points can be found in the own application executable, in its extensions, in its configuration parameters, both on startup or conditional, or in specific threat objects.

As in server software persistence points, the first category in user software ones is the own application or its extensions, regarding techniques consisting on their malicious manipulation in order to maintain persistence on a compromised system. Each time the compromised application is executed or its extensions are loaded by the user, the threat actor can execute malicious code on the target system. Naikon group is an example of an APT relying on these persistence points, as to maintain persistence it drops a malicious extension to be loaded by Microsoft Word at startup (Checkpoint, 2020). Examples of specific malware abusing these persistence points are Industroyer (Slowik, 2018; Di Pietro et al., 2021), which trojanizes Windows Notepad to establish a backdoor persistence mechanism, or Kobalos, which replaces the SSH client with a trojanized version in order to steal credentials on compromised systems (Léveillé and Sanmillan, 2021; Pleiter et al., 2021). In addition to the use of applications and their extensions as persistence points, a threat actor can rely on the particular configurations of these applications, both loaded at startup or under certain conditions. This approach is the same that we have stated in server software but, in this case, related to user software. For example, MuddyWater APT exploits Microsoft Office configurations to maintain persistence on a compromised target (TOK and CELİKTAS, 2019), while APT32 replaces Microsoft Outlook configuration files to implant a backdoor for persistence (Dahan, 2017). Finally, the last category inside user software persistence points, as in server software ones, is the one regarding malicious objects opened or loaded by an application. In this case, persistence is triggered when the threat object is accessed by the user. These persistence points are used regarding objects that are regularly accessed, even automatically loaded, by the user; in other case, persistence would be very weak for an advanced threat actor, as it would fully depend on the user manually opening a threat object. To our knowledge, this kind of threat objects without a guaranteed access are not commonly exploited, although it is technically possible. The only group we have identified relaying on these persistence points is Gamaredon, which inserts malicious macros into existing documents providing persistence when they are reopened (Boutin, 2020).

Please note that threat objects, as in server software, can be both local (for example, malicious templates to be loaded) and remote; in fact, not only threat objects, but also software configurations, extensions or even login related persistence points can be both local and remote. In these cases, as we did in OS dependent taxonomy, we consider again the concept of remote persistence point, as they are located outside the targeted system, so persistence relies on a third party, also compromised, system.

5.5. Summary

In our proposal we present a novel taxonomy for persistence points, those that store artifacts that are abused to maintain persistence in a compromised system. This model provides a direct taxonomy for techniques exploited by threat actors. We propose four high level families for these points: those regarding locations

Table 1
Persistence points proposed taxonomy.

	Hardware		
Pre–OS	Firmware		
	Boot device	Boot sector	
		Boot loader	
OS–dependent	Kernel	OS kernel	
		Modules	
	Boot procedures	Initialization scripts	
		Initialization databases	
		Startup items	
		Services	
		Environment variables	
	Accounts		
	Scheduled tasks	Periodic tasks	
		One–time tasks	
		Timers	
	Base software	Applications	
		Extensions	
		Configurations	Startup
			Conditional
		Threat objects	
	Remote points		
Server software	Applications		
	Extensions		
	Configurations	Startup	
		Conditional	
	Accounts		
	Threat objects		
User–dependent	Login related	Scripts	
		Items	
		Configurations	
	User software	Applications	
		Extensions	
		Configurations	Startup
			Conditional
		Threat objects	
	Remote points		

prior to the OS boot, those regarding locations directly linked to OS capabilities, those regarding locations related to server software as an addendum for the system and, finally, those related to user–related locations of the system. Each of them is divided into different families to provide an accurate persistence points taxonomy.

The proposed taxonomy for persistence points we have developed in this work is shown in Table 1, where we summarize the different persistence points families for all of the main stated categories.

6. Discussion

We have identified the absence of a suitable taxonomy for the techniques commonly used by advanced threat actors to achieve persistence. As with the rest of tactics MITRE ATT&CK defines, this framework provides a plain relationship for persistence techniques. Such a plain structure hinders the analysis and, most important, the detection, of these techniques. As persistence is mandatory for most advanced threat actor's operations, it is important to establish a suitable approach for persistence techniques that allows analysts to identify the persistence in a potentially compromised system.

Persistence is not only a key tactic for advanced threat actors, but also a key feature for malware. Persistence has been analyzed in three main research lines: Microsoft Windows techniques, specific malware capabilities and, during the last years, persistence in IoT–related infrastructures. None of these lines provides a suitable taxonomy for persistence techniques, but only weak classifications schemes for them, linked to specific operating systems or even malware capabilities. Without a global, platform–independent approach, a relevant problem for analysts is to identify persistence in environments that have not been previously explored.

In this paper we define the concept of persistence point as the location within a compromised system where a persistence artifact has been stored. Dealing with persistence as a global tactic for advanced threat actors, these locations are classified into a novel taxonomy for persistence points, thus establishing the locations of a system that have to be analyzed to identify persistence mechanisms. In this way, our approach provides a common reference for the identification of persistence techniques, as the relationship between persistence techniques and persistence points is direct. All persistence techniques rely on at least one persistence point. For analysts, by inspecting these points it is possible to detect the artifacts that achieve persistence, even when facing compromises in new environments or technologies.

To discuss the completeness and correctness of our work, we have mapped MITRE ATT&CK persistence techniques to our proposed taxonomy. This framework is the main public effort to establish a classification for tactics and techniques used by threat actors. As on May, 2022, MITRE ATT&CK "Persistence" tactic (last modified on 19th July 2019), identified as TA0003, consists of techniques that adversaries use to keep access to systems across restarts, changed credentials, and other interruptions that could cut off their access. MITRE ATT&CK provides no structure for techniques inside the "Persistence" tactic; the framework places all of these techniques at the same level, providing in some cases specific sub techniques. Although this approach is followed in all ATT&CK tactics and techniques, we advocate that it is important to provide a fine classification for all tactics, and in this case, for the "Persistence" one, by dividing its techniques at least in the first–level classification we provide in this work, based on persistence points.

The mapping of MITRE ATT&CK persistence techniques to the persistence points we propose in our work is shown in Table 2 .

As we can confirm, all techniques and subtechniques identified by MITRE ATT&CK for the "Persistence" tactic can be mapped to our proposed taxonomy. Based on persistence points, our approach provides not only this full coverage, but also a platform–agnostic structure for all these techniques, improving the detail that MITRE ATT&CK defines. Our structure also considers techniques not identified or partially identified in the MITRE ATT&CK framework: for example, our proposal extends T1137, Office Application Startup, to generic applications startup, considering not only Microsoft Office but any application a user can execute to achieve persistence.

Analyzing this mapping, it draws our attention that Hardware is a persistence point in the second classification level without MITRE ATT&CK associated techniques. This fact highlights the absence of techniques relying on hardware implants as persistence points. As we have stated in this work, these implants are expensive and highly platform–dependent, so threat actors do not use them on the wild. The other persistence point in the second classification level without linked techniques is server software extensions and configurations. In this case, this fact highlights that when a hostile actor uses server software to achieve persistence, it mostly relies on accounts, threat objects and even the own application binary as persistence points. This is a normal finding, particularly with the accounts and threat objects persistence points, as for a threat actor it is usually easier to abuse or manipulate such objects than to rely on extensions, not used in all server software deployments, or configurations, in many cases customized for each particular deployment and thus harder to iterate among multiple victims.

Another key finding is that Pre–OS persistence points are the less exploited ones by hostile actors. As we have stated in our work, although these persistence points are the hardest ones to detect and eradicate, their exploitation is usually expensive and difficult to achieve. In front of this situation, we can confirm that the abuse and manipulation of persistence points linked to the op-

Table 2
MITRE ATT & CK techniques mapping.

	Hardware	
Pre–OS	Firmware T1543.001 T1543.002 T1543.004	
	Boot device	Boot sector T1543.003
		Boot loader T1543.003
OS–dependent	Kernel	OS kernel T1547.006
		Modules T1547.006
	Boot procedures	Initialization scripts T1037.004
		Initialization databases T1547.001 T1547.003 T1547.004 T1547.005 T1547.010 T1037.001 T1546.001 T1546.002 T1546.007 T1546.008 T1546.009 T1546.010 T1546.011 T1546.012 T1546.015 T1574.011 T1137.002
		Startup items T1547.001 T1547.011 T1037.002 T1543.001 T1543.004
		Services T1097 T1547.002 T1547.008 T1547.010 T1547.012 T1543.002 T1543.003 T1546.003 T1546.014 T1574.010 T1053.004
		Environment variables T1574.001 T1574.004 T1574.006 T1574.007 T1574.012
	Accounts T1098.004 T1136.001 T1136.003 T1078.001 T1078.003 T1078.004	

(*continued on next page*)

Table 2 (*continued*)

	Hardware		
	Scheduled tasks	Periodic tasks T1053.003 T1053.005	
		One–time tasks T1053.001 T1053.002	
		Timers T1053.006	
	Base software T1205.001	Applications T1546.005 T1546.006 T1546.008 T1574.005	
		Extensions T1574.001	
		Configurations	Startup T1546.011 T1546.013 T1574.002 T1574.006
			Conditional
		Threat objects T1574.008 T1574.009	
	Remote points T1037.003 T1136.002 T1574.001 T1543.005	T1078.002	
Server software	Applications T1546.005 T1546.006		
	Extensions		
	Configurations	Startup	
		Conditional	
	Accounts T1098.001 T1098.002 T1098.003 T1133		
	Threat objects T1525 T1505.001 T1505.002 T1505.003		
User–dependent	Login related	Scripts T1546.004	
		Items T1547.001 T1547.007 T1547.009 T1547.011 T1543.001	
		Configurations T1098.004	
	User software	Applications T1554 T1546.005 T1546.006	

(*continued on next page*)

Table 2 (*continued*)

Hardware			
		Extensions T1176 T1137.001 T1137.006	
		Configurations	Startup T1574.002 T1137.003 T1137.004 T1137.005
			Conditional
		Threat objects T1574.008 T1574.009	
	Remote points T1137.004		

erating system boot procedures comprises most of the techniques from the framework. Particularly, an initialization database such as Windows Registry is the main persistence point for all analyzed MITRE ATT&CK techniques. Our taxonomy also extends this specific platform–dependent approach to a generic family of Initialization databases for OS–dependent persistence points.

Our approach significantly improves the analysis of persistence linked to specific operating systems or technologies. Platform–dependent approaches are only useful for the particular environments they are designed for, but they are not able to establish a common reference to be used in all platforms. None of the previous approaches we have analyzed defines a platform–agnostic classification, although some of them try to establish such a classification for the particular environments they study. These particular approaches have provided the common basis for our novel proposal of a general taxonomy suitable for all platforms. Our novel platform–agnostic taxonomy is a useful tool for defenders to face the detection of persistence techniques, as well as for the planning and execution of offensive operations such as cyberspace exploitation or cyberspace attack. The application of our proposal to both of these perspectives improves an organizations' security.

From a practical point of view, persistence detection is usually a complex task when facing advanced threat actors. These actors exploit uncommon techniques in a broad range of platforms, from standard operating systems such as Microsoft Windows or Linux to closed appliances, and even legacy systems. When facing incidents, MITRE ATT&CK framework is the common starting point for defenders. However, this framework is technology–dependent, so it can not be exploited to face persistence in new environments, and even to hunt not previously identified techniques in common platforms. In this sense, our taxonomy provides a general model that can be used to structure knowledge about persistence points, thus allowing analysts the identification of novel, or at least uncommon, ones. As all techniques rely on at least one persistence point, the identification of these persistence points directly implies the discovery of persistence techniques.

Relying on a closed framework for persistence analysis, we are limited to the previously identified persistence techniques. For example, MITRE ATT&CK identifies persistence techniques linked to different user software compromise; they include techniques linked to specific user software such as Microsoft Office or web browsers. Without a platform–agnostic model, persistence detection is mostly limited to these specific applications. By using our taxonomy, analysts can not only deal with persistence points linked to Microsoft Office or web browsers, but they can extrapolate them to other user applications, thus being able to identify new persistence points. Through this identification, security analysts can determine all the possible locations where an artifact can be stored for persistence purposes: i.e., the analysts are identifying where to look for, or where to implant (in an offensive operation), persistence artifacts. This is especially relevant not only in common platforms or technologies, but also when facing the identification of persistence techniques in novel, not previously analyzed, environments, such as legacy systems or proprietary appliances.

Once persistence points have been identified, from a defensive perspective these points must be analyzed in order to find traces of abuse or manipulation. Please note that, as we have previously stated, a persistence point can store not only malware, but also a full range of configurations or legitimate tools to enable persistence through them. This analysis can be performed through intrusion detection techniques, out of the scope of our proposal, such as misuse or anomaly detection. In addition, from an offensive perspective, the identification of persistence points will help the red team to determine the different points where an artifact can be stored in a target. Those points will range from the well–known ones to the less used ones, this is, to the less monitored ones. On a prior basis, the exploitation of uncommon persistence points will increase the probability of success for an offensive cyberspace operation.

Finally, in this section we identify machine learning as an especially interesting research line. Machine learning approaches can be applied not only to identify intrusions, including persistence, against infrastructure, but also to classify them into a suitable taxonomy of persistence points. Different researches have been conducted to analyze malware capabilities with machine learning approaches (Santos et al., 2013; Nath and Mehtre, 2014; Yuan, 2017; Bahtiyar et al., 2019). A summary of them can be found in Ucci et al. (2019) or Singh and Singh (2020). Nevertheless, although some of these approaches establish a suitable classification for malware (Gibert et al., 2020; Qamar et al., 2019), none has focused on the persistence mechanisms used in each case, neither in a potential classification for these mechanisms.

7. Conclusions

Persistence, the ability to keep presence in a targeted system for a long time, is a key tactic for the operations of advanced threat actors. These operations are expensive for an actor, so once a target has been compromised it is a common approach to keep control of this target as long as possible.

Persistence techniques have been largely analyzed in particular technologies, from specific operating systems to malware samples or ICS (Industrial Control Systems) environments. However, these analysis lack a global perspective, thus making it difficult to identify general capabilities that can be extrapolated from one environment to another. This is a relevant problem for security analysts when facing the potential compromise of new systems or technologies, as there is not a common reference to check for the identification of techniques in these environments.

In this work we provide a global platform–agnostic taxonomy for persistence techniques that allows the analysis of compromised systems regardless of their technology, thus easing the security analysts' work. This novel approach identifies persistence points as the locations within a system where persistence artifacts can be located. These points represent the components of a targeted infrastructure that are abused to maintain persistence, so they define the locations where analysts must check the presence of artifacts. The relationship between persistence points and persistence techniques is direct.

Our taxonomy is based on four main persistence points families, regarding those that are located before the OS boots, those that are located on OS–dependent capabilities, those that are located on server software and, finally, those that are user–dependent, mainly located in a user's particular files. All of these families are divided into different categories to specify where persistence artifacts can be located. As a first and novel taxonomy, our proposal can be used as a fundamental basis for new and more specific approaches.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Antonio Villalón-Huerta: Writing – original draft. **Hector Marco-Gisbert:** Writing – original draft. **Ismael Ripoll-Ripoll:** Writing – original draft.

References

Akbanov, M., Vassilakis, V.G., Logothetis, M.D., 2019. Wannacry ransomware: Analysis of infection, persistence, recovery prevention and propagation mechanisms. Journal of Telecommunications and Information Technology 113–124.

Axelrod, R., Iliev, R., 2014. Timing of cyber conflict. Proceedings of the National Academy of Sciences 111 (4), 1298–1303.

Bahrami, P.N., Dehghantanha, A., Dargahi, T., Parizi, R.M., Choo, K.-K.R., Javadi, H.H., 2019. Cyber kill chain-based taxonomy of advanced persistent threat actors: analogy of tactics, techniques, and procedures. Journal of information processing systems 15 (4), 865–889.

Bahtiyar, Ş., Yaman, M.B., Altniğne, C.Y., 2019. A multi-dimensional machine learning approach to predict advanced malware. Computer networks 160, 118–129.

Black, P., Gondal, I., Layton, R., 2018. A survey of similarities in banking malware behaviours. Computers & Security 77, 756–772.

Boutin, J.-I., 2020. Gamaredon group grows its game. Technical Report. ESET.

Breuk, R., Spruyt, A., 2012. Integrating dma attacks in exploitation frameworks. University of Amsterdam, Tech. Rep 2011–2012.

Brierley, C., Pont, J., Arief, B., Barnes, D.J., Hernandez-Castro, J.C., 2020. Persistence in linux-based iot malware. In: 25th Nordic Conference on Secure IT Systems (Nordsec).

Bytes, A., Zhou, J., 2020. Post-exploitation and persistence techniques against programmable logic controller. In: International Conference on Applied Cryptography and Network Security. Springer, pp. 255–273.

Calvet, J., Campos, J., Dupuy, T., 2016. Visiting The Bear Den. Technical Report. ESET.

Cayford, M., Van Gulijk, C., van Gelder, P., 2014. All swept up: An initial classification of nsa surveillance technology. Safety and Reliability: Methodology and Applications 643–650.

Checkpoint, 2020. Naikon APT: cyber espionage reloaded. Technical Report. Checkpoint.

Chen, Y.-H., Lin, Y.-D., Chen, C.-K., Lei, C.-L., Huang, C.-Y., 2020. Construct macos cyber range for red/blue teams. In: Proceedings of the 15th ACM Asia Conference on Computer and Communications Security, pp. 934–936.

CISA, 2020. Advanced Persistent Threat Compromise of Government Agencies, Critical Infrastructure, and Private Sector Organizations. Technical Report. Cybersecurity and Infrastructure Security Agency.

Cozzi, E., Graziano, M., Fratantonio, Y., Balzarotti, D., 2018. Understanding linux malware. In: 2018 IEEE symposium on security and privacy (SP). IEEE, pp. 161–175.

Dahan, A., 2017. Operation Cobalt Kitty: A large-scale APT in Asia carried out by the OceanLotus Group. Technical Report. Cyber Reason.

Di Pietro, R., Raponi, S., Caprolu, M., Cresci, S., 2021. Critical infrastructure. In: New Dimensions of Information Warfare. Springer, pp. 157–196.

Dupuy, T., Faou, M., 2021. Gelsemium. Technical Report. ESET.

Eder-Neuhauser, P., Zseby, T., Fabini, J., Vormayr, G., 2017. Cyber attack models for smart grid environments. Sustainable Energy, Grids and Networks 12, 10–29.

FBI/NSA, 2020. Russian GRU 85th GTsSS Deploys Previously Undisclosed Drovorub Malware. Technical Report. National Security Agency, Federal Bureau of Investigation.

FireEye, 2018. APT37 (REAPER). The Overlooked North Korean Actor. Technical Report. FireEye, Inc., Milpitas, CA, USA.

Gao, H., Li, Q., Zhu, Y., Wang, W., Zhou, L., 2012. Research on the working mechanism of bootkit. In: 2012 8th International Conference on Information Science and Digital Content Technology (ICIDT2012), Vol. 3. IEEE, pp. 476–479.

Gavaudan, L., Legras, S., Ventos, V., 2021. Cyber range automation, a bedrock for ai applications. Proceedings of the 28th C&ESAR 165.

Gibert, D., Mateu, C., Planes, J., 2020. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. Journal of Network and Computer Applications 153, 102526.

Gittins, Z., Soltys, M., 2020. Malware persistence mechanisms. 24th International Conference on Knowledge–Based and Intelligent Information & Engineering Systems 176, 88–97.

Grill, B., 2016. Bootkits revisited; detecting, analysing and mitigating bootkit threats. Technischen Universitat Wien.

Guri, M., Monitz, M., Elovici, Y., 2016. Usbee: Air-gap covert-channel via electromagnetic emission from usb. In: 2016 14th Annual Conference on Privacy, Security and Trust (PST). IEEE, pp. 264–268.

Hawley, S., Read, B., Brafman-Kittner, C., Fraser, N., Thompson, A., Rozhansky, Y., Yashar, S., 2019. APT39: An iranian cyber espionage group focused on personal information. Technical Report. Mandiant.

Hayashi, K., Ray, V., 2018. Bisonal Malware Used in Attacks Against Russia and South Korea. Technical Report. PaloAlto Networks.

Horejsi, J., Lunghi, D., Pernet, C., Kazuki, F., 2020. Earth akhlut: exploring the tools, tactics and procedures of an advanced threat actor operating a large infrastructure. VB 2020 localhost.

Hromcová, Z., 2019. At commands, TOR–based communications: meet Attor, a fantasy creature and also a spy platform. Technical Report. ESET.

Hwang, J.-h., Lee, T.-j., 2019. Study of static analysis and ensemble-based linux malware classification. Journal of the Korea Institute of Information Security & Cryptology 29 (6), 1327–1337.

Joy, J., John, A., Joy, J., 2011. Rootkit detection mechanism: a survey. In: International Conference on Parallel Distributed Computing Technologies and Applications. Springer, pp. 366–374.

Kamluk, V., Gostev, A., 2016. Adwind – A cross–platform RAT. Technical Report. Kaspersky.

Kao, D.-Y., Hsiao, S.-C., 2018. The dynamic analysis of wannacry ransomware. In: 2018 20th International Conference on Advanced Communication Technology (ICACT). IEEE, pp. 159–166.

Kirillov, I., Beck, D., Chase, P., Martin, R., 2011. Malware attribute enumeration and characterization. Technical Report.

Kramer, S., Bradfield, J.C., 2010. A general definition of malware. Journal in computer virology 6 (2), 105–114.

Kumar, S., et al., 2020. An emerging threat fileless malware: a survey and research challenges. Cybersecurity 3 (1), 1–12.

Kuraku, S., Kalla, D., 2020. Emotet malware'a banking credentials stealer. Iosr J. Comput. Eng 22, 31–41.

Lai, A.C.T., Wong, K.W.K., Wong, J.T.W., Lau, A.T.W., Ho, A.P.L., Wang, S., Muppala, J.K., 2021. Backdoor investigation and incident response: From zero to profit. In: 12th EAI International Conference on Digital Forensics and Cyber Crime.

LaSota, A., 2019. The present and potential future of mac hardware. implants.

Lee, G., Shim, S., Cho, B., Kim, T., Kim, K., 2020. Fileless cyberattacks: Analysis and classification. ETRI Journal.

Lemmou, Y., Lanet, J.-L., Souidi, E.M., 2021. A behavioural in-depth analysis of ransomware infection. IET Information Security 15 (1), 38–58.

Léveillé, M.-E.M., Sanmillan, I., 2021. A wild Kobalos appears. Tricksy Linux malware goes after HPCs. Technical Report. ESET.

Li, X., Wen, Y., Huang, M.H., Liu, Q., 2011. An overview of bootkit attacking approaches. In: 2011 Seventh International Conference on Mobile Ad-hoc and Sensor Networks. IEEE, pp. 428–431.

Mankin, J., 2013. Classification of malware persistence mechanisms using low-artifact disk instrumentation. Northeastern University, Boston, Massachusetts.

Matrosov, A., 2019. Uefi vulnerabilities classification focused on bios implant delivery. https://medium.com/@matrosov/uefi-vulnerabilities-classification-4897596e60af.

Matrosov, A., Rodionov, E., Bratus, S., 2019. Rootkits and bootkits: reversing modern malware and next generation threats. No Starch Press.

McGraw, G., Morrisett, G., 2000. Attacking malicious code: A report to the infosec research council. IEEE software 17 (5), 33–41.
Mishra, R., Jha, S.K., 2022. Survey on botnet detection techniques. In: Internet of Things and Its Applications. Springer, pp. 441–449.
Mohanta, A., Saldanha, A., 2020. Persistence mechanisms. In: Malware Analysis and Detection Engineering. Springer, pp. 213–236.
Monnappa, K.A., 2018. Learning Malware Analysis: Explore the concepts, tools, and techniques to analyze and investigate Windows malware. Packt Publishing Ltd.
Morgner, P., Pfennig, S., Salzner, D., Benenson, Z., 2018. Malicious iot implants: Tampering with serial communication over the internet. In: International Symposium on Research in Attacks, Intrusions, and Defenses. Springer, pp. 535–555.
Mwiki, H., Dargahi, T., Dehghantanha, A., Choo, K.-K.R., 2019. Analysis and triage of advanced hacking groups targeting western countries critical national infrastructure: Apt28, red october, and regin. In: Critical infrastructure security and resilience. Springer, pp. 221–244.
Nafisi, R., Lelli, A., 2021. Goldmax, goldfinder, and sibot: Analyzing NOBELIUM's layered persistence. https://www.microsoft.com/security/blog/2021/03/04/goldmax-goldfinder-sibot-analyzing-nobelium-malware/.
Namanya, A.P., Cullen, A., Awan, I.U., Disso, J.P., 2018. The world of malware: An overview. In: 2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud). IEEE, pp. 420–427.
Nath, H.V., Mehtre, B.M., 2014. Static malware analysis using machine learning methods. In: International Conference on Security in Computer Networks and Distributed Systems. Springer, pp. 440–450.
Németh, K., 2020. Detection of persistent rootkit components on embedded IoT devices.
Oosthoek, K., Doerr, C., 2019. Sok: Att&ck techniques and trends in windows malware. In: International Conference on Security and Privacy in Communication Systems. Springer, pp. 406–425.
O'Leary, M., 2019. Malware and persistence. In: Cyber Operations. Springer, pp. 507–566.
Pleiter, D., Varrette, S., Krishnasamy, E., Özdemir, E., Pilc, M., 2021. Security in an evolving European HPC Ecosystem. Technical Report. PRACE aisbl.
Popli, N.K., Girdhar, A., 2019. Behavioural analysis of recent ransomwares and prediction of future attacks by polymorphic and metamorphic ransomware. In: Computational Intelligence: Theories, Applications and Future Directions-Volume II. Springer, pp. 65–80.
Qamar, A., Karim, A., Chang, V., 2019. Mobile malware attacks: Review, taxonomy & future directions. Future Generation Computer Systems 97, 887–909.
Ramaswamy, A., 2008. Detecting kernel rootkits. Technical Report. Dartmouth College, NH, USA.
Rao, H., Selvakumar, S., 2014. A kernel space solution for the detection of android bootkit. In: ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society of India-Vol I. Springer, pp. 703–711.
Riley, R., Jiang, X., Xu, D., 2008. Guest-transparent prevention of kernel rootkits with vmm-based memory shadowing. In: International Workshop on Recent Advances in Intrusion Detection. Springer, pp. 1–20.
Russinovich, M.E., Margosis, A., 2016. Troubleshooting with the Windows Sysinternals Tools. Microsoft Press.
Russinovich, M.E., Solomon, D.A., Ionescu, A., 2009. Windows® Internals. O'Reilly Media, Inc.
Sanjay, B., Rakshith, D., Akash, R., Hegde, V.V., 2018. An approach to detect fileless malware and defend its evasive mechanisms. In: 2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS). IEEE, pp. 234–239.
Santos, I., Devesa, J., Brezo, F., Nieves, J., Bringas, P.G., 2013. Opem: A static-dynamic approach for machine-learning-based malware detection. In: International joint conference CISIS'12-ICEUTE' 12-SOCO' 12 special sessions. Springer, pp. 271–280.
Sharma, A., Gandotra, E., Bansal, D., Gupta, D., 2019. Malware capability assessment using fuzzy logic. Cybernetics and Systems 50 (4), 323–338.
Sikorski, M., Honig, A., 2012. Practical malware analysis: the hands-on guide to dissecting malicious software. No Starch Press.
Singh, J., Singh, J., 2020. A survey on machine learning-based malware detection in executable files. Journal of Systems Architecture 101861.
Slowik, J., 2018. Anatomy of an attack: Detecting and defeating crashoverride. Virus-Bulletin.
Slowik, J., 2019. Evolution of ICS attacks and the prospects for future disruptive events. Technical Report. Threat Intelligence Centre Dragos Inc, Hanover, MD, USA.
Stewin, P., Bystrov, I., 2012. Understanding dma malware. In: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, pp. 21–41.
Thompson, E.C., 2020. Threat intelligence. In: Designing a HIPAA-Compliant Security Operations Center. Springer, pp. 37–63.
TOK, M.S., CELİKTAS, B., 2019. Muddywater apt group and a methodology proposal for macro malware analysis. Bilişim Teknolojileri Dergisi 12 (3), 253–263.
Ucci, D., Aniello, L., Baldoni, R., 2019. Survey of machine learning techniques for malware analysis. Computers & Security 81, 123–147.
Uroz, D., Rodríguez, R.J., 2019. Characteristics and detectability of windows auto-start extensibility points in memory forensics. Digital Investigation 28, S95–S104.
Ussath, M., Jaeger, D., Cheng, F., Meinel, C., 2016. Advanced persistent threats: Behind the scenes. In: 2016 Annual Conference on Information Science and Systems (CISS). IEEE, pp. 181–186.
Varlioglu, S., Elsayed, N., ElSayed, Z., Ozer, M., 2022. The dangerous combo: Fileless malware and cryptojacking. In: IEEE Region 3 Technical, Professional and Student Conference.
Vogl, S., Pfoh, J., Kittel, T., Eckert, C., 2014. Persistent data-only malware: Function hooks without code. NDSS. Citeseer.
Wakabayashi, S., Maruyama, S., Mori, T., Goto, S., Kinugawa, M., Hayashi, Y.-i., 2017. Poster: Is active electromagnetic side-channel attack practical? In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 2587–2589.
Wang, Y.-M., Roussev, R., Verbowski, C., Johnson, A., Wu, M.-W., Huang, Y., Kuo, S.-Y., 2004. Gatekeeper: Monitoring auto-start extensibility points (aseps) for spyware management. In: LISA, Vol. 4, pp. 33–46.
Wardle, P., 2014. Invading the core: Iworm's infection vector and persistence mechanism. Virus Bulletin.
Wardle, P., 2014. Methods of malware persistence on mac os x. In: Proceedings of the virus bulletin conference.
Webb, M.S., 2018. Evaluating tool based automated malware analysis through persistence mechanism detection. Kansas State University.
Wei, F., Li, Y., Roy, S., Ou, X., Zhou, W., 2017. Deep ground truth analysis of current android malware. In: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, pp. 252–276.
Yuan, X., 2017. Phd forum: Deep learning-based real-time malware detection with multi-stage analysis. In: 2017 IEEE International Conference on Smart Computing (SMARTCOMP). IEEE, pp. 1–2.
Zimba, A., Chishimba, M., 2017. Exploitation of DNS tunneling for optimization of data exfiltration in malware-free APT intrusions. Zambia ICT Journal 1 (1), 51–56.
Zimba, A., Wang, Z., 2017. Malware-free intrusions: Exploitation of built-in pre-authentication services for APT attack vectors. International Journal of Computer Network and Information Security 9 (7), 1.

Antonio Villalón-Huerta is Chief Security Officer at S2 Grupo. He holds a MSc in Computer Engineering from the Universidad Politecnica de Valencia, Spain. With 25 years of experience in the cyber security field, in his career he has executed and managed analysis, defense, attack and exploitation projects, as well as designed and managed security operations and incident response centers. Antonio is the author of different books, articles and chapters on the subjects of cyber security and cyber intelligence, as well as a regular speaker in many congresses and courses. His research interests include the Russian cyber intelligence community and the modeling and detection of advanced threat actors.

Dr. H. Marco-Gisbert (M'13-SM'18) is an associate professor and cybersecurity researcher at the Universitat Politecnica de Valencia, Spain. He holds a PhD in Computer Science, Cybersecurity, from Universitat Politecnica de Valencia. Hector is senior member of the Institute of Electrical and Electronics (IEEE), and member of the Engineering and Physical Sciences Research Council (EP- SRC) in UK. Previously, he was associate professor at University of the West of Scotland, UK and cybersecurity researcher at the Universitat Politecnica de Valencia where he co-founded the "cybersecurity research group". Hector was part of the team developing the multi-processor version of the XtratuM hypervisor to be used by the European Space Agency in its space crafts. He participated in multiple research projects as Principal Investigator and Co-Investigator. Hector is author of many papers of computer security and cloud computing. He has been invited multiple times to reputed cybersecurity conferences such as Black Hat and DeepSec. Hector has published more than 10 Common Vulnerabilities and Exposures (CVE) affecting important software such as the Linux kernel. He has received honors and awards from Google, Packet Storm Security and IBM for his security contributions to the design and implementation of the Linux ASLR. Hector's professional interests include low level cybersecurity, secure and non-secure world kernel and userland security, virtualization security and applied cryptography.

Ismael Ripoll-Ripoll received the PhD in computer science from the Universitat Politécnica de Valéncia in 1996, where he is currently professor of several cybersecurity subjects in the Department of Computing Engineering. In reverse chronological order: before working on security, he participated in multiple research projects related to hypervisor solutions for European spacecrafts; dynamic memory allocation algorithms; Real-Time Linux; and hard real-time scheduling theory. Currently, he is applying all this background to the security field. His current research interests include memory error defense/attacks techniques (SSP and ASLR) and software diversication. He is the leader of the Cybersecurity researcher group at the UPV.