Unveiling stealth attack paths in Windows Environments using AWARE

Manuel Poisson, Valérie Viet Triem Tong, Gilles Guette, Erwan Abgrall, Frédéric Guihéry, Damien Crémilleux

Unveiling stealth attack paths in Windows Environments using AWARE

Manuel Poisson
Amossys, CentraleSupélec, CNRS, Inria, Univ. Rennes, IRISA
manuel.poisson@irisa.fr

Valérie Viet Triem Tong
CentraleSupélec, CNRS, Inria, Univ. Rennes, IRISA
Valerie.VietTriemTong@centralesupelec.fr

Gilles Guette
Univ. Rennes, CNRS, Inria, IRISA
gilles.guette@univ-rennes.fr

Erwan Abgrall
CentraleSupélec, Inria
erwan.abgrall@centralesupelec.fr

Frédéric Guihéry
Amossys
frederic.guihery@amossys.fr

Damien Crémilleux
Amossys
damien.cremilleux@amossys.fr

***Abstract*—When an attacker targets a system, he aims to remain undetected as long as possible. He must therefore avoid performing actions that are characteristic of an identified malicious behavior. One way to avoid detection is to only perform actions on the system that appear legitimate. That is, actions that are allowed because of the system configuration or actions that are possible by diverting the use of legitimate services. This article presents and experiments AWARE (Attacks in Windows Architectures REvealed), a defensive tool able to query a Windows system and build a directed graph highlighting possible stealthily attack paths that an attacker could use during the propagation phase of an attack campaign. These attack paths only rely on legitimate system actions and the use of Living-Off-The-Land binaries. AWARE also proposes a range of corrective measures to prevent these attack paths.**

***Index Terms*—Attack graph, attack campaign, intrusion, LotL.**

I. Introduction

An attacker and a defender engage in an enduring adversarial interaction. Classically, a defender uses a monitoring solution and/or security tools to observe the actions made on the system and detect if the actions in progress could be symptomatic of an attack. The defender's tools can implement signature-based approaches, which consist of verifying in real time that observations made on the system exhibit characteristics of known attacks. The attacker easily goes undetected by employing unknown attack techniques or new variations of already known techniques. This approach can be costly for the attacker because it requires him to constantly renew his attack tools and procedures. Another possible approach for attackers is to use only legitimate tools that already exist on the target system, but to abuse their usage in order to progress in the attack campaign. One evasive tactic that has become popular among attackers and malware authors is the usage of Living-Off-The-Land (LotL) techniques [1]. These techniques rely on the use of onboard tools that are specific to the operating system and that are used out of their regular scope to perform malicious actions on a host system. These tools can therefore be used in all stages of the killchain [2]: from the initial compromise to the effect on the target.

A major problem of the use of LotL by the attacker, is that these tools can hardly be removed from the system because they are mandatory for its legitimate usage and administration. This article details how AWARE (Attacks in Windows Architectures REvealed) computes the set of attack positions that an attacker can reach in a system and how the attacker can progress by exploiting only legitimate privileges and tools. AWARE also proposes countermeasures to break the identified attack paths. These countermeasures are ordered according to the impact the remediation may have on the system.

The article is organized as follows. Section II details the related works, Section III explains the construction of the attack positions graph, Section IV details the possible remediations to identified attack paths. Finally, Section V experiments AWARE on multiple information systems.

II. Related work

Active Directory (AD) is arguably the most popular solution for organizations to manage and organize their staff's IT profiles [3]. AD is used for identity management and allows a central authentication/authorization mechanism in an information system relying on Windows OS. It is then a privileged target for cyber-attacks [4]. Secure management and administration of Active Directory architectures is a complex task and some tools have been developed to audit such architectures [5]. These tools can be used to defend and protect information systems by revealing existing vulnerabilities so that they can be fixed prior to any attack. Audit tools like Oradad [6], PingCastle [7] and Bloodhound [8] are specifically designed to audit architectures that use AD. Some of them only collect data on domain controllers (DC), computers used to manage other resources. This concerns information needed for the entire information system. Others also collect local data that is only

available on client workstations, like local groups members, by sending requests to all known machines in the network.

In an architecture using AD, Oradad [6] grabs knowledge about users, groups and computers, among others, by sending LDAP queries to domain controllers. Nevertheless, data is collected in the form of text files that are very dense and tedious to exploit. PingCastle [7] goes further than Oradad by retrieving data from both DC and also some local data, only stored on client workstations. PingCastle then generates a report containing vulnerabilities found in the analyzed architecture. Each vulnerability comes with a security score and a description containing links to external references. To find vulnerabilities, various configurations of the architecture are checked against AD good practices. PingCastle can also generate a graph with nodes representing some AD objects and edges displaying their logical relationships, like the membership of a user to a group. Edges are colored following an AD health check score when available. This graph may help to understand the global structure of an AD architecture, but it cannot be used to easily find attack paths like in AWARE's graph. Bloodhound [8] aims at revealing (potentially unintended) relationships between various AD objects represented as nodes in a graph. It can be used to find an insecure path between an unprivileged user and a DC. Attack paths are better represented with the Bloodhound graph than with the PingCastle one since the former shows the possibility for users to execute actions on remote machines (with RDP or PSRemote for example) while PingCastle's graph does not. Adalanche [9], similarly to Bloodhound, allows to discover some attack paths in an AD. As opposed to AWARE's graph, the semantic of two nodes can differ in the heterogeneous graphs generated by these tools.

The scope of these tools is AD only. These first approaches miss local configurations (as local groups and local users) and are also unaware of the real use of the system and the resulting vulnerabilities In contrast, AWARE combines both AD and local data. On the other hand, other tools focus on local privileges escalation only. For instance, Mimikatz [10] allows, on certain conditions, to dump credentials saved on a machine. In the same way, WinPEAS [11] checks if one or multiple ways to elevate his privileges exist on a machine using the Windows OS. Among others, it allows detection of weak configurations on services. These tools highlight a probably vulnerable configuration on a machine, or host, but do not explain how it can be exploited. They are only able to unveil the fact that one user could take control of another one. However, it is the succession of movements from one user account to another that are the most difficult to detect and, therefore, the most dangerous. Also, a machine in an architecture is often connected to others. That must be remembered because an attacker gaining access to a new user may be only a step toward accessing a new machine. AWARE builds an attack paths graph that aims to highlight available combinations of movements between users and machines in an architecture.

III. Modeling paths in a compromised network

We consider a network where an advanced attacker has already succeeded in an initial compromise and therefore has access to at least one legitimate account on a machine. We are interested in the so-called *propagation phase* in which the attacker will progress in the network to reach his target.

This article details how AWARE interrogates the target system to compute the set of attack positions that an attacker is likely to reach and then the set of stealthy paths between these different attack positions. An attack position is a pair $(machine, user)$. This term of *attack position* was originally proposed in [12] and simply refers to an attacker who has compromised a $user$ account on a $machine$. The propagation space of an attacker is the set of machines and accounts that he is able to discover and control during the propagation phase. This propagation space can be modeled using a graph of attack positions. This graph has directed labeled edges to model the progression during the attack. The attacker can indeed move from one attack position to another with:

- lateral movements: the attacker takes control of another machine while keeping the same user account
- vertical movements (*aka* privileges escalations): the attacker takes control of another user account while remaining on the same machine.

To perform a lateral or vertical movement, the attacker can use multiple attack techniques. Many of these are listed in the Mitre ATT&CK matrix under the tactics `privileges escalation`[1] and `lateral movement`[2].

In this article, we focus on the stealthy movements that an attacker could achieve thanks to Living-Off-The-Land (LotL) techniques. LotL refers to binaries already present on systems (*e.g.* signed and legitimate administration tools) to conduct post-exploitation activity. The LOLBAS [3] project lists Microsoft-signed legitimate files that present extra functionalities which can be abused by attackers to conduct their attack campaign. This repository results from a large-scale systematic investigation of the use of these techniques by malwares [1]. The LotL techniques allow an attacker to remain stealthy. Their usage is very difficult to detect because it uses only legitimate actions, allowed by the system. The results in [1] confirm low detection rates for several documented LotL techniques for almost every popular anti-virus product tested.

The computation of the propagation space by AWARE is both static and dynamic. AWARE consults the system configuration to determine the possible attack positions and a part of the edges. This graph is then enriched by the edges made possible by the real use of the system in production.

A. Computing attack positions in a Windows system

In a pure Windows system, the set of possible attack positions for each machine is the union of the set of users

[1] https://attack.mitre.org/tactics/TA0004/
[2] https://attack.mitre.org/tactics/TA0008/
[3] https://lolbas-project.github.io/#

who can log on locally to the machine with the set of the ones who can log on remotely to that same machine.

A Windows system is classically governed by one or several Active Directory Domain Services (AD) in charge of authorizing and authenticating users and computers in a Windows domain type network. The AD, therefore, defines the set of possible attack positions on the system, except for the purely local accounts on the machines. The recommended way, to define if a user can log on a machine in AD is to set the user logon rights which allows to set these rights at the same time on many users (groups) for many computers thanks to group policy objects (GPOs). For each machine in an AD, the user right `SeInteractiveLogonRight`, respectively `SeDenyInteractiveLogonRight`, defines a list of users allowed, respectively forbidden, to physically open a session on the machine. These rights can be re-defined using GPOs.

The right `Se(Deny)RemoteInteractiveLogonRight` is the counterpart of `Se(Deny)InteractiveLogonRight` to allow remote connections using Remote Desktop Protocol (RDP). It can also be defined using GPOs.

Besides, it is also possible to define if a user can log on a computer by setting his attribute `User-Workstations`. This must be done user by user and computer by computer, which is fastidious and prone to errors. Therefore, it is rarely used and Microsoft documentation notes that it should not be used anymore while recommending to use `Se(Deny)InteractiveLogonRight` as explained above. By default, the user attribute `User-Workstations` is undefined and has no effect.

AWARE queries the AD to build the set of attack positions, *i.e.* all the pairs (m, u) where u is a user having an account on a machine m. All the users including purely local users are considered as able to access all the machines (including virtual machines) except those explicitly denied. The two rights `SeDenyInteractiveLogonRight` and `SeInteractiveLogonRight` define the ability to log in physically on a machine. If the two are contradictory, the first has priority over the second. Then, if the right `SeInteractiveLogonRight` is not granted or if the right `SeDenyInteractiveLogonRight` is granted for u on m the attack position (m, u) is not considered by AWARE.

B. Directed graph of attack positions

At this stage, the constructed graph contains only nodes that are all the attack positions an attacker could reach. In this work, we are interested in attack paths using only legitimate actions that an advanced attacker could use to remain stealthy. We begin to present how to add edges that are first due to access rights and then, we focus on edges that are due to the misuse of legitimate services.

An attacker can switch from one attack position to another using the Remote Desktop Protocol (RDP) and PowerShell Remote (PSRemote).

a) RDP: is a communication protocol that allows a user to connect to a remote machine and use its graphical interface as if he was in front of the machine. A user u can connect to a remote machine m if, on that machine m if he is a member of the local group `Remote Desktop Users` or `Administrators` and if he has the right `SeRemoteInteractiveLogonRight` but does not have the right `SeDenyRemoteInteractiveLogonRight`. By default, `SeRemoteInteractiveLogonRight` is given to the local group `Administrators`. On client workstations (*i.e.* not domain controllers) the local group `Remote Desktop Users` also has it by default. No user has the right `SeDenyRemoteInteractiveLogonRight` by default. These rights can be re-defined using GPOs. The machine initiating the connection must use `mstsc.exe` (Microsoft Terminal Service Client) and the target of the connection must use and activate an RDP server software. These software are present, whilst deactivated, by default on any machine using Windows. The firewall of the machine that is the target of the connection must allow incoming traffic for RDP, which common port is 3389.

b) PSRemote: is a feature of PowerShell that allows a user to execute commands and scripts on a remote machine. PSRemote uses the Windows Remote Management (WinRM) service and the PowerShell Remoting Protocol to establish a connection to the remote machine and execute commands. For example, a system administrator could use PSRemote to remotely update a set of machines in his network. The right to use PSRemote can be defined by GPOs. For instance, a GPO (`Allow remote server management through WinRM`) defines, for a given machine, the IP range of source machines that are allowed to use PSRemote to access it. Also, the firewall must allow incoming traffic from the IP that will use PSRemote on the target machines. Targets of PSRemote connections use a WinRM listener exploiting HTTP(S). This listener is present by default on Windows machines. To be allowed to use PSRemote on a machine, a user must be a member of the local group `Administrators` or `Remote Management Users` of this machine. By default, this latter group is empty for all machines.

AWARE interrogates the AD to know the rights granted to the users to use RDP and PSRemote and then adds the corresponding edges in the graph of the attack positions.

Finally, an attacker can abuse existing services on a machine that is accessible from his attack position. We found 22 LotL tools related to one of the Mitre ATT&CK tactics `execution`, `privilege escalation`, `credential access` or `lateral movement`. Among these, 40% are taken into account by AWARE and detailed in Table I. Due to space limitation, we detail only the usage of `wmic` and `runas` and more generally the abuse of `Modifiable service executable`.

c) WMIC: is a command-line interface to use WMI, a set of tools used to manage Windows systems. The program `wmic` is classified as a LotL since it allows to execute arbitrary commands on a remote machine. A user is allowed to run commands through `wmic` on a target m if, on m, he has specific permissions on the WMI namespace `root/cimv2` and if he is a member of the local group `Performance Log Users`,

TABLE I
LOTL TOOLS ALLOWING TO STEALTHILY PROGRESS IN AN ARCHITECTURE

Name	Movement	Usage
sc	vertical & lateral	To run arbitrary executables with the SYSTEM account rights on a remote machine, create and start a service with a custom binpath on a remote machine or modify binpath of an existing service
msiexec	vertical	To run arbitrary commands with the SYSTEM account rights, use Microsoft System Installer to execute arbitrary commands while "installing" a custom .msi file
winrm	vertical & lateral	To run arbitrary executables (with the SYSTEM account rights) (on a remote machine), create/start a service with a custom binpath (on remote machine) or create process on (remote) machine
runas and cmdkey	vertical	Discover name of a new user and run arbitrary commands with any user account (possibly without knowing his password)
netsh	vertical & lateral	To run arbitrary dll files (with the SYSTEM account rights) (on a remote machine), add a helper (*i.e.* a .dll file) to the binary netsh.exe that will be executed when netsh.exe runs
dnscmd	vertical	To run arbitrary dll files (with the SYSTEM account rights) on DC, add a plugin (*i.e.* a dll file) to the DNS service
eventvwr	vertical	To run arbitrary executables with local admin rights, bypass UAC when running the executable as local admin by setting custom executable path as the value of registry `HKCU_Software_Classes_mscfile_shell_open_command`
schtasks and at	lateral	To run arbitrary executables, schedule a task that will be executed on remote machine (with the rights of the specified user)
wmic	lateral	To run arbitrary commands (executed with the rights of the specified user), start a new process on a remote machine

`Distributed COM Users` or `Administrators`.

d) RUNAS: is a Windows built-in command that allows a user to run a command on behalf of another user. The command `runas /user:u\dom cmd.exe`, for example, starts by asking for the password of the user u (member of the AD domain `dom`). If and only if the correct password is given, a new command prompt is opened with the account of u, no matter who ran the `runas` command. However, it is possible to skip the part that asks for the password and to use the saved credentials of any user, when they are available. This can be convenient to avoid typing a password again and again when a user needs to run multiple commands on behalf of another one. If the password of u is saved, simply adding the argument `/savecred` to the `runas` command allows to run any command on behalf of u, without even knowing his password, which can be exploited by an attacker.

e) Modifiable service executable: On Windows machines, a service is a program running in the background. There are hundreds of services to provide various functionalities. When a service starts running (classically when the machine is booted), it executes a file with the rights of a specific account. Many default services use the rights of `NT AUTHORITY/SYSTEM` which has the highest privileges on the local machine.

A user that have the permission to rewrite the file that is executed by a service can replace it with an arbitrary file. When the service is restarted, the new file gets executed with a possibly very privileged account. Indeed, services are regularly running with `NT AUTHORITY/SYSTEM` rights. This insecure and dangerous configuration can be exploited by attackers to perform a vertical movement by executing commands with the rights of a new user (possibly `NT AUTHORITY/SYSTEM`).

AWARE queries all machines in the architecture from a single machine that is part of the analyzed AD to take into account movements from one attack position to another allowed by abusing existing legitimate services. For each running machine, AWARE collects services names, the path of their executable and access control lists on these paths, the right access on the root WMI namespace. Besides, on each machine, saved credentials are checked.

The process to know if any user has any saved credentials on any machine is as follows. For each machine m in the network, we define a user u who has saved the credentials of a user u' on m . These credentials are written $cred_{m,u,u'}$. They are saved in the encrypted file $f_{m,u,u'}$ that can be decrypted using $mkey_{m,u}$. Here, $mkey_{m,u'}$ represents the masterkey used to decrypt $f_{m,u,u'}$ and get all the credentials the user u saved on m. We have $cred_{m,u,u'} = D(f_{m,u,u'}, mkey_{m,u})$, where D represents the decryption process. The masterkey is also stored in an encrypted file $f'_{m,u}$ that can be decrypted using the key $backupK$. We have $mkey_{m,u} = D'(f'_{m,u}, backupK)$. $backupK$ is stored on the domain controller (DC) and can be used to decrypt masterkeys from all machines. Considering data collection is made from machine m_0, for each machine and for each user, the files $f_{m,u,u'}$ and $f'_{m,u}$ are copied from m to m_0. Then $backupK$ is also copied from the DC to m_0. Finally, $cred_{m,u,u'}$ is found using $cred_{m,u,u'} = D(f_{m,u,u'}, \underbrace{D'(f'_{m,u}, backupK)}_{=mkey_{m,u}})$. The login and password of the user u' are contained in $cred_{m,u,u'}$. The only information stored by AWARE is the tuple $(m, u_{login}, u'_{login})$ because nothing more is needed to know that u can move vertically to u' on machine m. For confidentiality purposes, all passwords and all files and keys used in the decryption process are automatically deleted before the script ends.

IV. CRITICAL PATHS IDENTIFICATION AND REMEDIATIONS

Attack paths, or critical paths, are paths from a node representing an initial attack position to a critical node, which is the attacker's target. Once the directed graph of stealth attack paths is created, it is simple to search for paths between two nodes given as parameters using graphs theory and well-established path-finding algorithms. The choice of these nodes is left to the users of AWARE. The entry node depends on

the security assumptions made on the information system. For example, AWARE allows to study the propagation space corresponding to an attacker who successfully compromised a particular machine/user or attack position. AWARE also allows calculating the set of initial attack positions allowing to reach a particularly sensitive position. These sensitive positions should be identified during the risk analysis. They could be, for instance, positions involving DCs or users with high privileges.

Finally, AWARE proposes remediations in the form of system configuration updates to secure an information system where critical paths have been identified. Remediations application change properties, settings and access rights in the information system to make it safer by removing critical paths. For example, to remove an edge labeled with `runas`, it is possible to simply delete the saved credentials for the user involved on the specific machine related to this edge. For an edge labeled with `PSRemote`, either PSRemote can be disabled on the target machine of the connection or the rights to use PSRemote on the machine can be removed for the user involved. For each type of edge, AWARE can propose at least one remediation to remove the edge and therefore break the killchain [2] before the attacker can reach his final goal.

One remediation can have a broader impact than removing a single edge in the attack paths graph. A single remediation can remove multiple critical paths at once. It will also probably remove edges that are not part of any attack path or even nodes in the graph. In the latter case, it means that the remediation removed the ability for a user to log into some machines. This type of undesirable side effect can negatively impact legitimate needs in the information system.

For each available remediation for a particular critical path, AWARE computes a score. This score captures the impact of the remediation on the system and is equal to the cost of the remediation minus its gain. The cost is equal to the number of accounts on machines that have to be deleted, updated or created plus the number of rights or services that have to be removed or updated which corresponds to the number attack positions and edges removed or updated from the graph. The gain corresponds to the number of critical attack paths successfully deleted by the remediation application.

AWARE proposes a list of remediations ordered by increasing scores to remove the critical path allowing to choose the best compromise.

V. Experiments

AWARE has been evaluated on 4 different information systems relying on AD. `LAB` is our testbed, it is fully virtual and available here[4]. `B` is a cyber-range used for training in cyber-attack/defense. `A` and `C`, are copies of AD used in production in a medium (`A`) and a large (`C`) organization.

For each analyzed information system, the number of users and machines, the execution times and the composition of the resulting graphs are detailed in Table II.

[4]https://gitlab.inria.fr/mpoisson/aware-lab-data

[5]median execution time in minutes

TABLE II
Experiments results

Name	nb users	nb machines	collect data[5]	build graph[5]	nb nodes	nb edges
LAB	22	6	8.2	0.0	55	92
A	146	123	missing	0.0	2830	1731
B	627	160	183.9	2.0	65265	13468
C	13100	154	26.0	152.1	1989253	31752

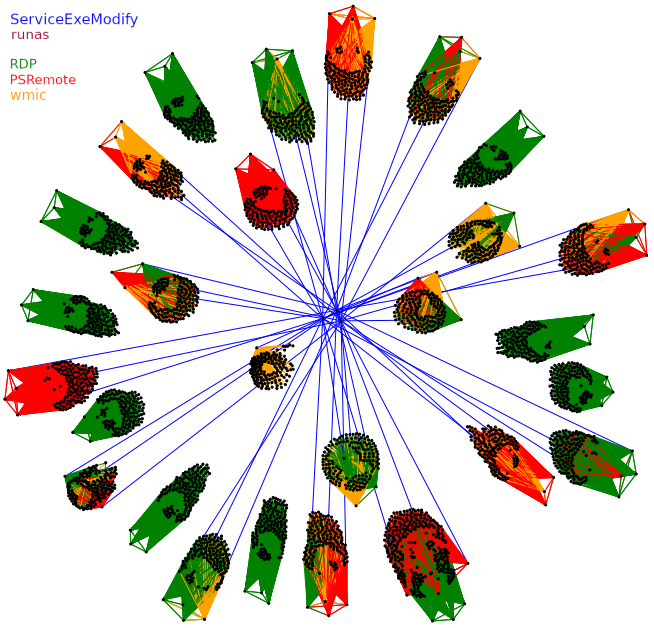

Fig. 1. Attack paths graph from `C`

a) `C` graph analysis: The experiment on `C` enhances the ability of AWARE to analyze large information systems in a reasonable time even when the generated graph contains about 2 million nodes and 30000 edges. The large size of `C` is likely to be the one of real enterprise information systems. Their complexity enhances the need for tools like AWARE to automatically analyze their security. Data collection on realistic and uncontrolled information systems was challenging and some data could not be collected. For instance, it concerns, for `C`, the saved credentials, members of local groups for some machines and some user attributes. This is due to some missing rights of the user executing AWARE's data-collector and to some impossible remote connections due to the network configuration. In addition, some machines were turned off during data collection and, therefore, could not answer requests from AWARE. On the one hand, uncollected restrictive rights could lead to false attack positions in the graph that do not exist in reality. On the other hand, uncollected data like saved credentials could lead to an exploitable vertical movement that is not represented by the graph. The attack paths graph of `C`, depicted in Fig. 1, shows clusters of nodes (or attack positions) representing the ability of the same user to perform lateral movements from many machines to a few others where he has additional privileges. At the center,

nodes involving the user *system* on different machines can be reached with `ServiceExeModify` edges (in blue), only by a few other nodes involving user accounts dedicated to service management. This graph did not unveil particularly critical paths. This is reassuring concerning the security of `C` but could be qualified by the fact that `C` is a pre-production information system, without real activity on it. This last aspect could justify that no saved credentials have been found on `C`.

b) `B` graph analysis: The shape of the graph from the information system `B` is similar to the one of `C`. Defining as critical the nodes involving DCs or users members of highly privileged groups (like `DOMAIN ADMINS`), all source nodes of paths reaching these critical nodes also were critical nodes. In `B`, only critical nodes could reach other critical nodes.

c) `A` graph analysis: On `A`, the attack paths graph shows a `runas` edge between nodes $(espark, admin)$ and $(espark, john)$. It reveals that on the machine $espark$, the user $admin$ can run arbitrary commands as if he was the user $john$ without knowing his password. Considering the data collected on `A`, another interesting fact, which increases the risk of this information system, is that the group `Authenticated Users` is a member of the local group `Administrators` on machine m_{12}. The group `Authenticated Users` has been found empty on `A`. However, it is reasonable to imagine the addition of an unprivileged user u_{added} to this group. In that case, u_{added} will also become a member of the local group `Administrators` on machine m_{12}. It will allow him to perform different lateral movements (*i.e.* remote connections) to this machine. AWARE allowed to check that the addition of u_{added} and lead to a new graph with new edges from nodes (m, u_{added}) to nodes (m_{12}, u_{added}) to represent the new lateral movements of u_{added} from various machines m to m_{12}.

d) `LAB` graph analysis: The full attack paths graph of the `LAB` information system is depicted in Fig. 2 and is available here[6]. The graph is made of two distinct, unconnected parts. At the bottom right, 4 nodes represent isolated attack positions that cannot be reached from any other attack position and that cannot lead to any new attack position.

The graph unveils the shortest path in `LAB` to gain access to the highly privileged user $system$ on machine $wksuni$, from the initial access of $(wksuni, pilipip)$. This path is highlighted at the bottom left of Fig. 2. It corresponds to an attack scenario where an attacker uses `RDP` from $(wksuni, pilipip)$ to reach the node $(wksop, pilipip)$. Here, he uses `runas` to reach $(wksop, sayanel)$ and then access $(wksuni, sayanel)$ with `wmic`. From there, he reaches his goal, $(wksuni, system)$, thanks to `ServiceExeModify`.

In the experimental information system `LAB`, we can imagine a scenario where the unprivileged user $pilipip$ has been compromised. The node $(wksuni, system)$ can be defined as a critical attack position since it involves the user $system$ on the machine $wksuni$. AWARE will therefore unveil the attack path from $(wksuni, pilipip)$ to $(wksuni, system)$ and it will propose a list of the best remediations to apply to break this

[6]https://gitlab.inria.fr/mpoisson/aware-lab-data

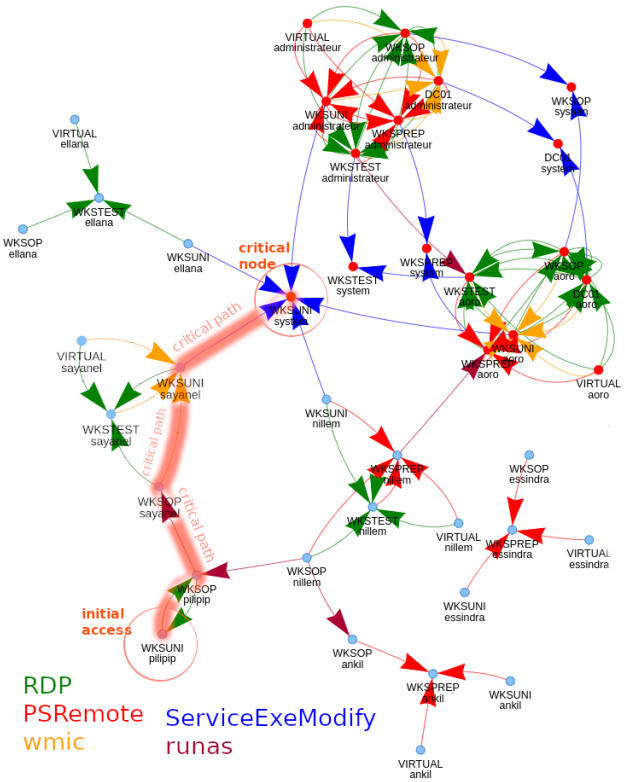

Fig. 2. Attack paths graph from `LAB`

path. In total, 6 remediations are possible to remove this attack path composed of 4 edges. For example, a possible way to remove the edge `wmic` is to apply the remediation associated with the action "On computer $wkstest$ remove user $sayanel$ from group $GSCI@LAB$". It deletes 1 node and 47 edges, and it updates 5 edges and removes 2 critical paths. Therefore, it has the lowest score compared to the other remediations available and AWARE will sort it as the first one to propose.

Experiments show that AWARE is able to generate an attack paths graph from information systems of various sizes and natures. The graph allows to easily visualize possible movements between users and machines in the information system. Thus, AWARE unveils existing critical paths that attackers could follow and proposes the least impactful remediations that remove a maximum number of critical paths.

VI. Conclusion

The knowledge of propagation paths in the targeted system is important information. Some attack paths exist in the system because the attacker is able to exploit vulnerabilities in the code of the services offered by the system. If the supervision system is up-to-date, the defender should be able to identify the exploitation of these vulnerabilities (while waiting to correct them). On the other hand, if the attacker is able to abuse the configuration of the system and legitimate services to propagate, then he becomes almost undetectable.

In this article, we have presented AWARE (Attacks in Windows Architectures REvealed), a tool for defender teams.

AWARE makes the defender aware of the propagation space of an attacker by generating a directed graph of attack positions. In the AWARE attack graph, moving from one position to another is possible without using advanced attack procedures. On the contrary, it only relies on the (ab)use of access rights or Living-Off-The-Land functions and services. We experimented with 4 realistic information systems of different sizes and natures. The generated graphs allowed to verify if attack paths existed. AWARE was then used to propose an ordered list of remediations to help in the removal of the unveiled attack paths while having a minimal impact on the analyzed system.

We plan to work on a better scoring system for remediations to improve the relevance of the remediations proposed by AWARE. An idea would be to let the user define some legitimate connections that must remain intact to keep the information system functional and make sure the proposed remediation do not remove these connections.

REFERENCES

[1] F. Barr-Smith, X. Ugarte-Pedrero, M. Graziano, R. Spolaor, and I. Martinovic, "Survivalism: Systematic analysis of windows malware living-off-the-land," in *2021 IEEE Symposium on Security and Privacy (SP)*, 2021, pp. 1557–1574.

[2] A. Berady, V. Viet Triem Tong, G. Guette, C. Bidan, and G. Carat, "Modeling the Operational Phases of APT Campaigns," in *CSCI 2019 - 6th Annual Conf. on Computational Science & Computational Intelligence*, Las Vegas, United States, Dec. 2019, pp. 1–6. [Online]. Available: https://hal.inria.fr/hal-02379869

[3] G. McDonald, P. Papadopoulos, N. Pitropakis, J. Ahmad, and W. J. Buchanan, "Ransomware: Analysing the impact on windows active directory domain services," *Sensors*, vol. 22, no. 3, 2022.

[4] W. Matsuda, M. Fujimoto, and T. Mitsunaga, "Detecting apt attacks against active directory using machine leaning," in *2018 IEEE Conference on Application, Information and Network Security (AINS)*, 2018, pp. 60–65.

[5] T. Loret, "Active Directory : Comparaison de différents outils d'audit et d'entretien," Nov. 2021. [Online]. Available: https://evabssi.com/active-directory-comparaison-de-differents-outils-daudit-et-dentretien/

[6] "ORADAD," Dec. 2022, original-date: 2018-12-18T13:27:49Z. [Online]. Available: https://github.com/ANSSI-FR/ORADAD

[7] "Pingcastle-home." [Online]. Available: https://www.pingcastle.com/

[8] "BloodHound: Six Degrees of Domain Admin — BloodHound 4.2.0 documentation." [Online]. Available: https://bloodhound.readthedocs.io/en/latest/

[9] L. Karlslund, "Adalanche Open Source," Jan. 2023, original-date: 2020-10-07T10:07:22Z. [Online]. Available: https://github.com/lkarlslund/Adalanche

[10] S. DUCKWALL and B. Delpy, "Abusing microsoft kerberos: Sorry you guys don't get it," *Blackhat*, 2014.

[11] E. Caroscio, J. Paul, J. Murray, and S. Bhunia, "Analyzing the ransomware attack on d.c. metropolitan police department by babuk," in *2022 IEEE International Systems Conference (SysCon)*, 2022, pp. 1–8.

[12] A. Berady, M. Jaume, V. Viet Triem Tong, and G. Guette, "PWNJUTSU: A dataset and a semantics-driven approach to retrace attack campaigns," *IEEE Transactions on Network and Service Management*, pp. 1–13, 2022. [Online]. Available: https://hal.inria.fr/hal-03694719