Complexity of Active Directory Attack Graphs

Arpit Gole
arpit.gole@student.adelaide.edu.au

Hung Nguyen
hung.nguyen@adelaide.edu.au

1 Introduction

Microsoft Active Directory and Azure Active Directory are directory services products used for identity and access management at most major enterprises all over the world. Since this attack primitive was first studied in 2009 [6], attackers are increasingly using this attack vector with great success by leveraging the intricacies of Active Directory (AD) to penetrate the environment through an exponential number of attack paths, offering virtually unlimited options for lateral movements. A recent study by Enterprise Management Associates [12] reports that 50% of businesses experienced an attack on their Active Directory systems in 2020-2021, with over 40% indicating the attack was successful.

AD systems are very complex. A medium enterprise network with thousands of users could have hundreds of thousands of security principals and millions of inter-dependencies between these principals that need to be managed. In most cases, these inter-dependencies are hidden in obscure access control settings that are not visible. Because AD systems are so complex, small security lapses can lead to gaping holes over time. Misconfigurations are a common occurrence in AD environments. Typical misconfigurations and poor security practices include letting third-party service accounts run with high privileges, privileged users accessing nonprivileged systems, and excessive permissive rights to normal users. Any of these security lapses when exploited by attackers with the right level of skills can lead to domain administrator privilege from user's privilege. The danger of misconfiguration problems is that there is no simple patch. Organisations apply temporary fixes such as access control audits and accounts reviews. Over time these band-aid solutions build up and misconfigurations do not go away but multiply and are pushed deeper inside the networks.

Culture and behavioural issues also contribute to AD problems. AD isn't always exclusively owned by the security team. AD is first and foremost an infrastructure technology. Many applications and other components depend on AD and as such AD may be built and owned by an infrastructure and operations team that may not have security concerns as a top priority. This means security teams will have to engage and convince the reluctant operations team to get their tools into the system and their security recommendations are implemented. Human errors and shortcuts contribute to compound the problem.

AD systems are dynamic and continuously evolve over a period of time. People's roles and access requirements change within an organisation. A well-executed access control process is required to manage the constant change requests on AD permissions. Security concerns not only apply to employees who are currently at the organisation but also to those who leave. If their credentials are not removed or expired, at best they become hidden and unexploited vulnerabilities and at worst they could be a powerful tool for both disgruntled former employees, as well as attackers who gain access to their credentials.

Companies don't have the tools and human expertise to understand and address AD security problems. Most companies have a poor understanding and visibility of their AD system. Simple questions such as the number of domain controllers, the number of domains and people with administrator privileges in Active Directory, and the overall hierarchy are beyond the reach of many network administrators. This lack of visibility has led to cases where IT administrators don't even understand the implications of their actions. They may see no harm in granting one user access to a machine, but an attacker with the same access could cause significant damage. A seemingly innocent step, combined with the misconfigurations in AD, can lead to complete compromise of the enterprise network.

2 Background and Related Work

Organisations lose track of the AD graph original schema when it grows over time. Administrators perform a variety of operations on AD - create/manage users, resource access management, groups and their access control and many more. For a long time now, many researchers have tried to figure out ways and techniques to make the task of AD maintenance a hassle-free undertaking by providing the maximum level of security with it. Once we have an AD running for many years now, guaranteeing the safety of the AD is a challenging task as the attack paths are not the only way the attackers tend to gain access to the network. Attackers also tend to leverage legitimate accounts, methods like using logs with machine learning, and it can detect the paths for the cases when the legitimate accounts are compromised [10].

As the complexity of the AD graph increases, so does the total number of attack paths. Some of the well-known attack paths are Golden/Silver Ticket, Kerberoasting, DCSync, Pass the ticket, Pass the Hash and DCShadow. The legacy technique(s) of protecting AD and doing a manual audit of AD user(s) are not sufficient for newer forms of challenges - the use of the sophisticated hacking tool and integration of hybrid environments [3]. Due to this, one of the widely used software is SIEM to detect and stop attackers from exploiting the well-known attack paths in the AD graph [8].

Graph-based analysis techniques [7, 9]are being aimed at identifying/solving vulnerabilities of the AD, which otherwise are very hard to detect - lateral movements. Hence, attack path(s) gets exploited by an attacker to gain access to the AD. The reality is they are hard to detect and even harder to build intuition around them, and they exist in every AD network of a sufficiently large organisation. Also, there is no way to stop an attacker if he initiates an attack after gaining access through one of these attack paths. Hence, it becomes critical to find and remove all the attack paths in an AD, where the core issue with this is the complexity of the schema of AD stacks up as the AD graph evolves.

3 Problem Definition

What is complexity? General complexity is an area of science itself (see for example https://en.wikipedia.org/wiki/Complexity). There are several research institutes dedicated to the study of complexity such as the Santa Fe institute in the US. There are many definitions of complexity that are used in different branches of science and depending on the specific applications. A few of these include:

a. Komogorov complexity
b. Chaotic
c. Information complexity
d. Computational Complexity

We focus here on a narrow definition of complexity that is directly related to network security and in particular how they impact on the ability of network administrators to understand the security issues of their network. In particular, the question that most network engineers do not have access to and often severely impact their capability to build a secure system:

"How does a local configuration changes (connectivity of a nodes and privilege of a user account) affect the overall system security?"

To answer this question, we need concrete definitions for two key important terms:

1. Local changes: We define here changes that occur to both *horizontal* and *vertical* changes. Horizontal changes happen when a physical nodes (hosts, computers) changes its connectivity with its neighbours. Mechanisms for these changes are discussed in the next section. Vertical changes occur when an account (users or services) status is changed. These could be privilege upgrade/down grade or account disabled. We again explain these dynamics in the next section.

2. System security: In AD security, the foremost security question is if an attacker compromises a user account, how easy it is for him to perform a snowball attack and compromise the higher privilege accounts such as domain admins, etc? To measure this system security, we will use the following metrics:

   a. Number of Users have path to Domain Admins.
   b. Percentage of Users have paths to Domain Admins.
   c. Average number of steps from Users to the DA (not counting unreachable nodes)
   d. Average number of Paths from a User to DA (not counting unreachable nodes)
   e. Number of Computers have paths to Domain Admins
   f. Percentage of Computers have paths to Domain Admins
   g. Average number of steps from Computer to the DA (not counting unreachable nodes)
   h. Average number of Paths from a Computer to DA (not counting unreachable nodes)

Note, that we also have generic graph metrics that help to keep track of the size of the overall graphs as follows:

a. Number of Hosts
b. Number of Users
c. Number of GPOs
d. Number of Domains
e. Number of Groups
f. Number of OUs
g. Number of CanRDP relations
h. Number of GpLink relations
i. Number of AdminTo relations
j. Number of GetChanges relations
k. Number of Contains relations
l. Number of WriteDacl relations
m. Number of MemberOf relations
n. Number of GenericAll relations
o. Number of GenericWrite relations
p. Number of Owns relations
q. Number of AddMember relations
r. Number of WriteOwner relations
s. Number of HasSession relations
t. Number of TrustedBy relations
u. Number of GetChangesAll relations
v. Number of AllowedToDelegate relations
w. Number of ForceChangePassword relations
x. Number of AllExtendedRights relations

4 Model

The generation of AD graph comprises 2 main steps - generating a base AD graph and evolving the AD graph for the specified number of teaching cycles.

4.1 Base Model

We mimic a University AD system for the purpose of illustration. First, the architecture of the AD graph reflects the separations of concerns [4, 5, 11].The primary identification factor for the different types of nodes in the AD graph is how it relates to a faculty - the graph uses *faculties* as a parameter - a list of names of faculties within a university to maintain the record. This architectural schema keeps the AD graph intuitive with the terminologies/limitations used in university. One such important parameter throughout the generation process is *num_nodes* - which governs the creation of several different types of nodes. Also, due to the high frequency of operations at fixed regular intervals (teaching cycles), we decided to keep AD as a centralized administrative model, everything under a single domain.

Other critical details related to generating the base AD graph process, i.e. assumptions made during the generation process of different types of Resources/Members, Organisational Unit (OU), Group Policy Object (GPO), various relationship types and all the various groups, are all governed by mathematical formulations.

Computers and servers are the 2 types of resources available in the AD graph. The total number of computers in the university is equal to the *num_nodes*. The count of servers is limited to 1/3rd of the total number of computers. In addition, each faculty contains its domain controller. An individual in an AD graph can either be a Staff member or a student (User). The total number of students enrolled in the university is again equal to the *num_nodes*. The Student/Faculty ratio is 3 - the total number of staff members are limited to 1/3rd of the total number of users.

Each faculty gets 2 main OUs - **_RESOURCES* and **_MEMBERS*. The total number of users, staff members, servers and computers are equally divided into different faculties during the generation process.Newly formed groups are then nested into other groups, staff members and users. Staff members are associated with a group with lower variance than users get associated with a group from the maximum possible connections. To govern the working environment of the AD graph, at max 20 GPOs are created. Each GPO in the graph gets connected with a minimum of 1 and a maximum of 3 OUs.

Several different relationship types govern the linking process throughout the architecture of the complete AD graph. Minimum of 1% and a maximum of 5% of users or staff members respectively are made Domain Admins. 5% of total staff have admin rights to a minimum of 1 and a maximum of 5% of the computers. Whereas for the servers maximum limit is 3% of the total servers. Secondly, 5% of the users also have admin rights to a minimum of 1 and a maximum of 5% of the computers. Finally, certain groups (IT groups) acts as admins to a minimum of 1 and a maximum of 5% of the computers. Some IT groups (10% of total IT groups) are sought after and have admin access to 30% of the computers.

All computers are accessible by users/staff members but, servers are only accessible by staff members. A user/staff member can have a session on 0 computers or at max the logarithmic value of total computers. Server sessions are limited to staff members based on the same formulation but with servers. Only certain users and groups (IT users and IT groups) combined with computer and staff members can have CanRDP, ExecuteDCOM and AllowedToDelegate access to 10% of the total computer. But only 5% of servers are governed by staff members using these relationship types.

Finally, unrestricted access to the complete AD graph is given only to Domain admins.

4.2 Evolution Model

Each evolution cycle mimics a teaching period of a university, i.e. a semester. The evolution cycle tries to replicate the daily process of an organisation in the time steps of a teaching cycle. The parameter *time_period* controls the number of evolution cycles to run. The resultant AD graph serves as the base graph for the next evolution cycle. We determined the following procedure to grow the base graph with each evolution step.

As we know, only a few computers demand a service after each semester. So, the script chooses only 20% of the time to service a c number of computers. If the script decides to service computers - it deletes all the relationship(s) and removes the computer from the AD graph. In place of the removed computer, it creates a new computer, accessible to use and makes relevant connections within the graph.

Each faculty have users connected. The script mimics the graduation process of students - it passes 20% of users from each faculty during each evolution cycle. Also, mimics the subsequent intake of new students to the faculties. The graduated users from faculties get entirely removed from the AD graph. The graduated user's entire history gets deleted from the AD graph, i.e. computer history, groups, etc. Also, the script creates some Kerberoastable users - to allow the possibility of Kerberoasting.

About 80% of the existing groups get pruned from the AD graph. The entire history related to the dropped group gets deleted. After this, new groups are generated. Then gets connected to other groups, staff members and users with the same assumptions as discussed in the above section.

4.3 Implementation

We used the Python programming language in the development of the code. The base AD graph gets created by a custom script highly influenced by the DBcreator script [1]. An entirely different script takes care of the evolution cycle(s).

We also used the Bloodhound [2] tool to introspect the construction of the AD graph. The bloodhound tool provided the much-needed analysis gateway to the AD graph. It pinpointed the areas to focus on, which attackers may use for malicious intentions. The graph data that we generated from the scripts combined with the bloodhound tool we were able to access and query the AD graph to discern the Attack paths.

Finally, the above clearly illustrates the 2 main steps - generating a base AD graph and evolving the AD graph.

5 Analysis

5.1 Simulation settings

For our simulations, we kept the values of parameters - *faculties* as a list of 7 different names, each identifying a faculty, *num_nodes* as 500, *c* as 20 and *time_period* as 1. Firstly, we generated a base graph in Run1. Then, we ran the evolution script 9 times and collected data for each run.

5.2 Findings

In basic terms, complexity can be viewed as the total number of objects (Members, Resources, Groups, etc.) are present in the AD graph and how they evolve over time. Figure 1 shows the nodes increasing over time, i.e. Run 1 to Run 10.

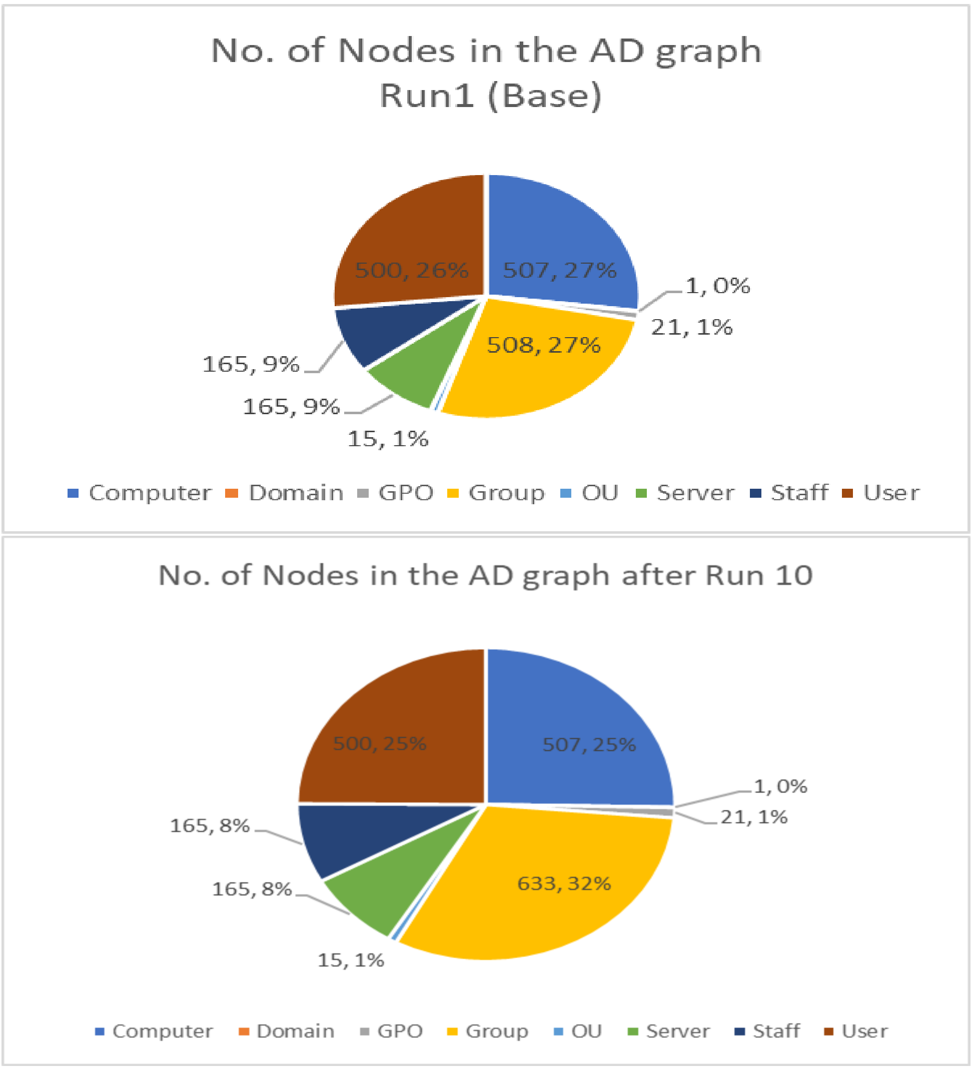
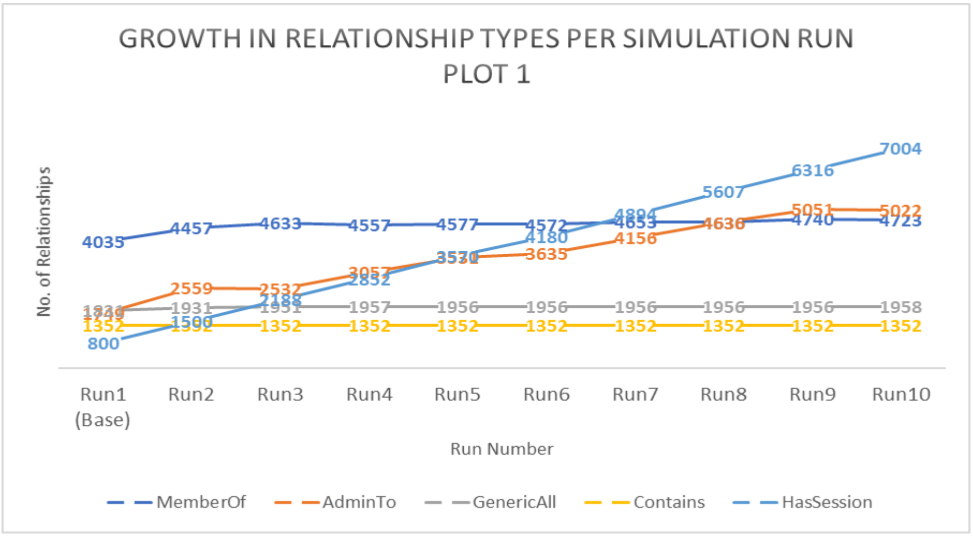


Figure 1: variation in number of nodes in the AD graph

The number of users, staff members, computers and servers are kept constant, whereas the number of groups increases. With this, it increases the connections within the AD graph shown below.



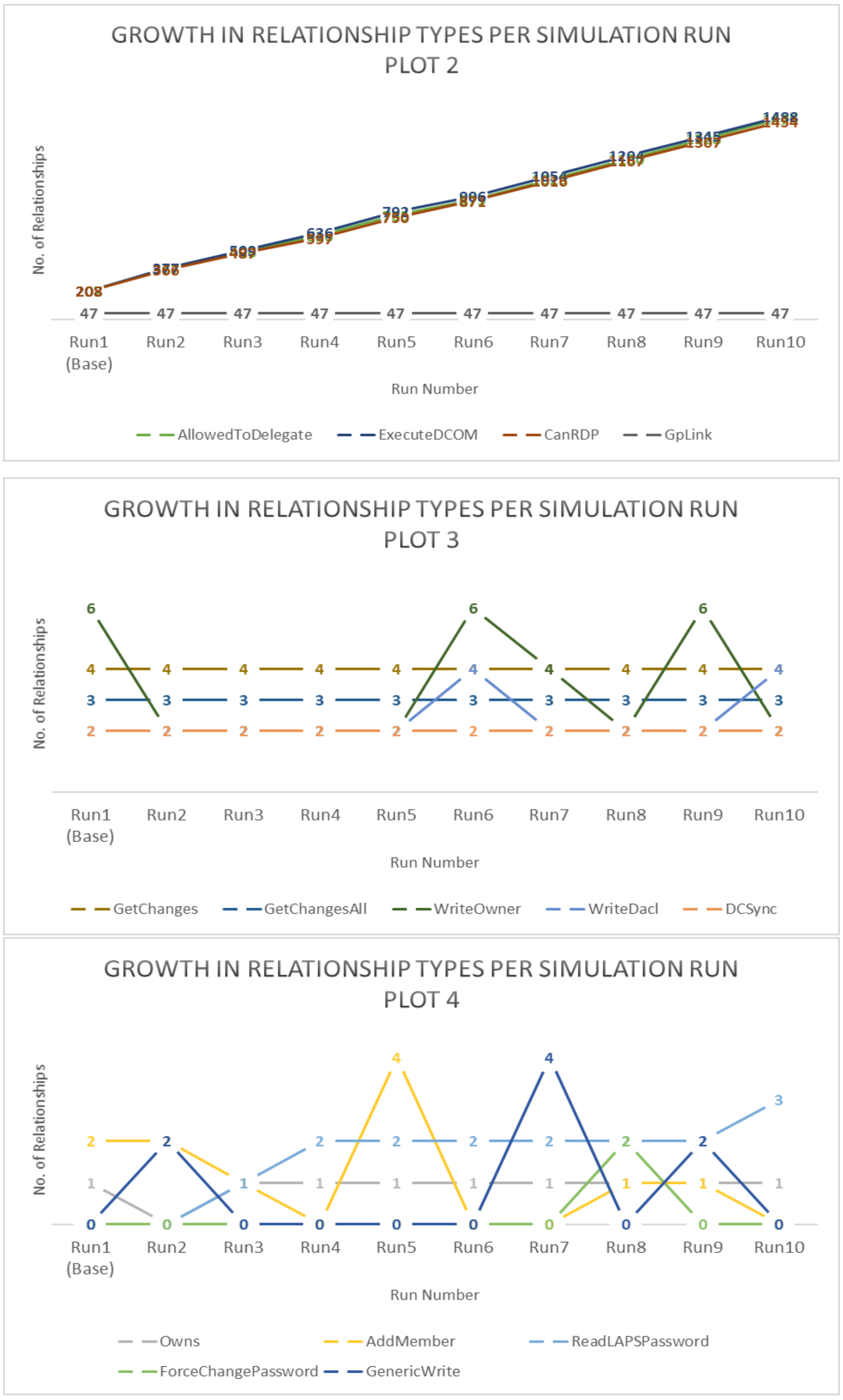


Figure 2: Growth in Relationship Types

Figure 2 above makes it clear that the number of connections increases even within a controlled environment. Some relationship types are rare whereas some like *Contains* are constant due to the fact the number of nodes is kept constant. Otherwise, it also got the potential to show a positive linear trend. As clearly shown by many other relationship types like *AdminTo*, *HasSession*, *ExecuteDCOM* and *CanRDP*. This trend can be explained, as these relationship types relate to the interaction connections.

Hence, it is evident that with each evolution cycle, the interaction within the AD increases.

The no. of attack paths from a user or computer to the Domain Admins (DAs) increases with each evolution cycle. Also, the number of computers contributing to the number of short attack paths are much higher than the number of users. As clearly shown in Figure 3, since the old computers in the AD graph have many unsecured *HasSession* relationships still active from the usage of old users.

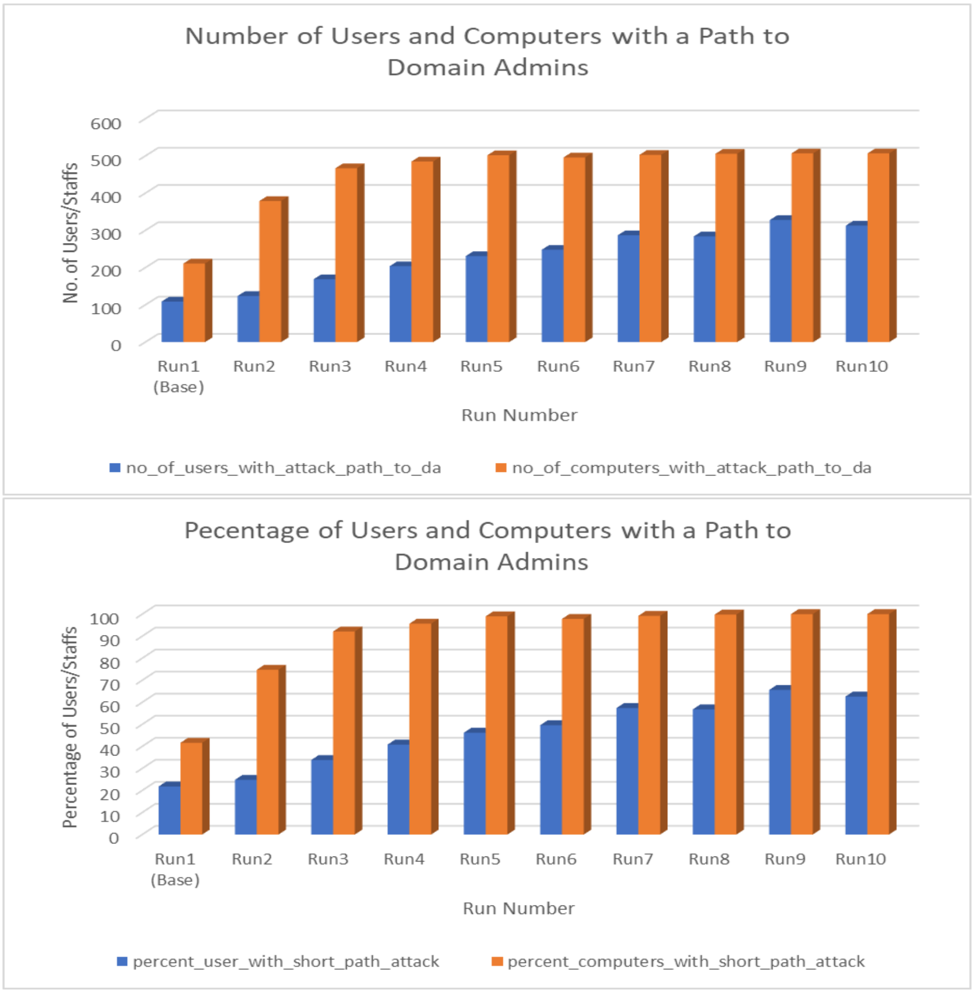


Figure 3: Increase in Attack paths

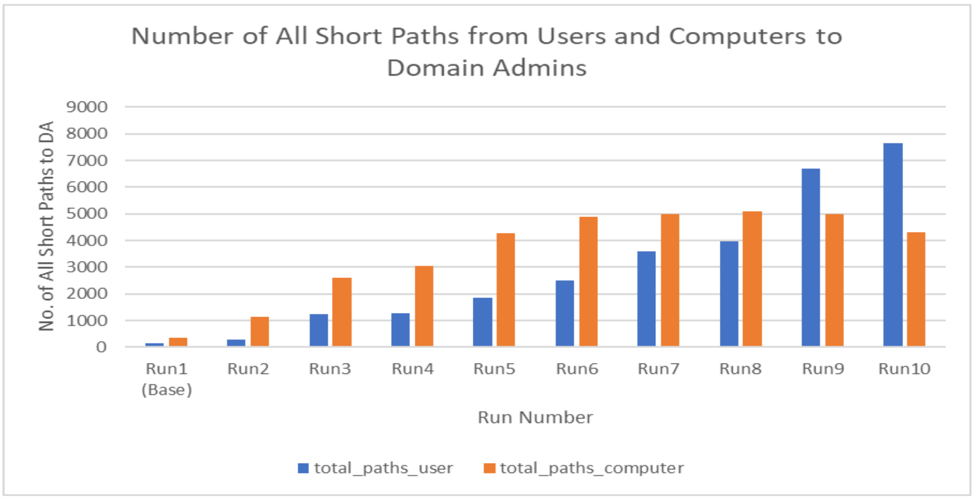
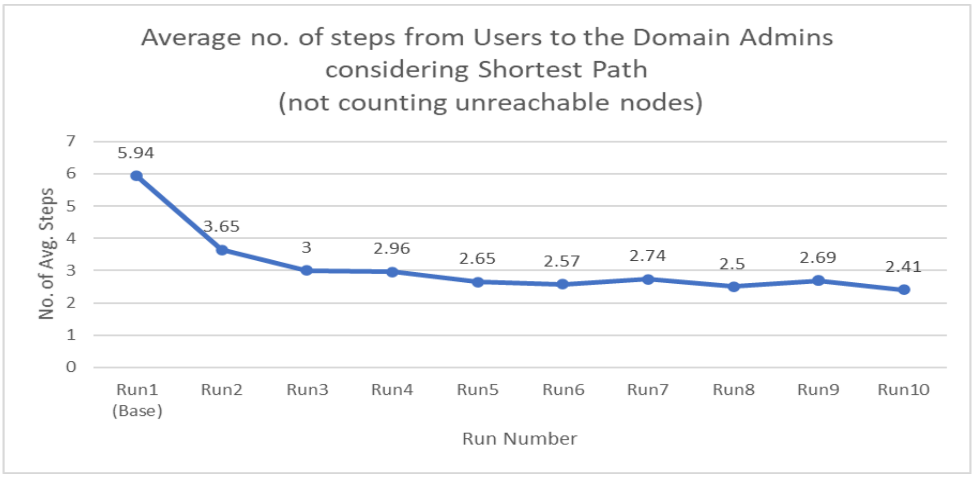


Figure 4: All Short Attack Paths from Computers and Users

Fewer users contribute to the attack paths compared to computers from Figure 3. Still, the number of all attack paths from users to DAs is much higher when compared to all attack paths from computers to DAs. Figure 4 depicts this behaviour because the users have sessions on many computers and are linked to different groups.



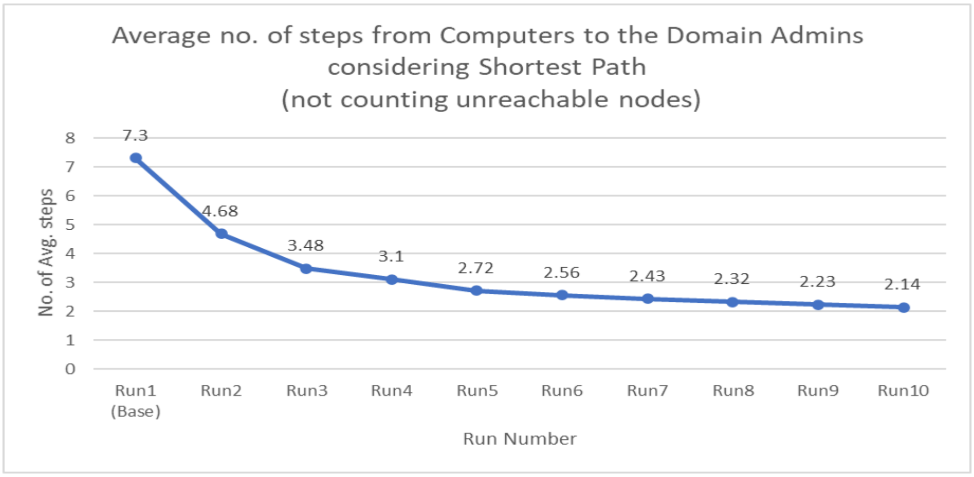
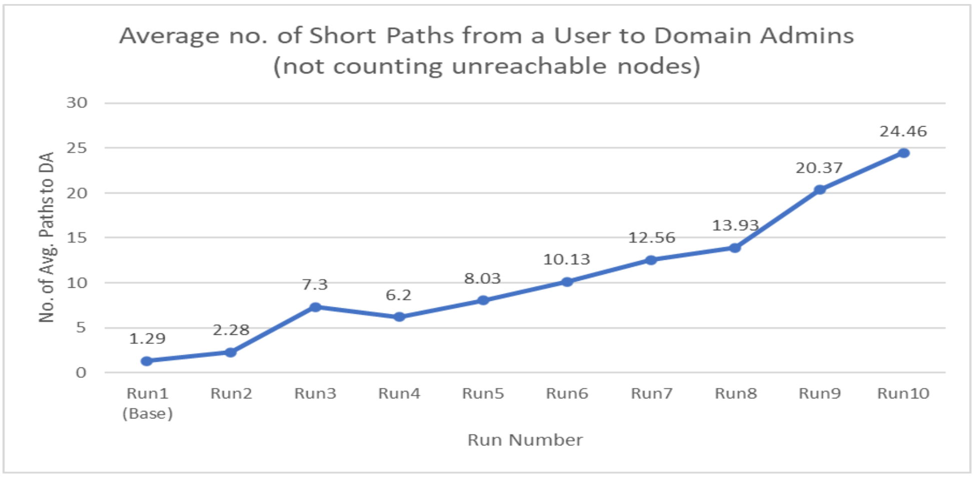


Figure 5: Avg. no. of steps from a Computer/User to DA
(based on a shortest path)

The average number of steps required from a user/computer to DAs decreases with each evolution cycle, as shown in Figure 5. Indicating more and more attack paths are increasing. Thus, in turn, it means it's easier to compromise the DAs and the whole AD in general.



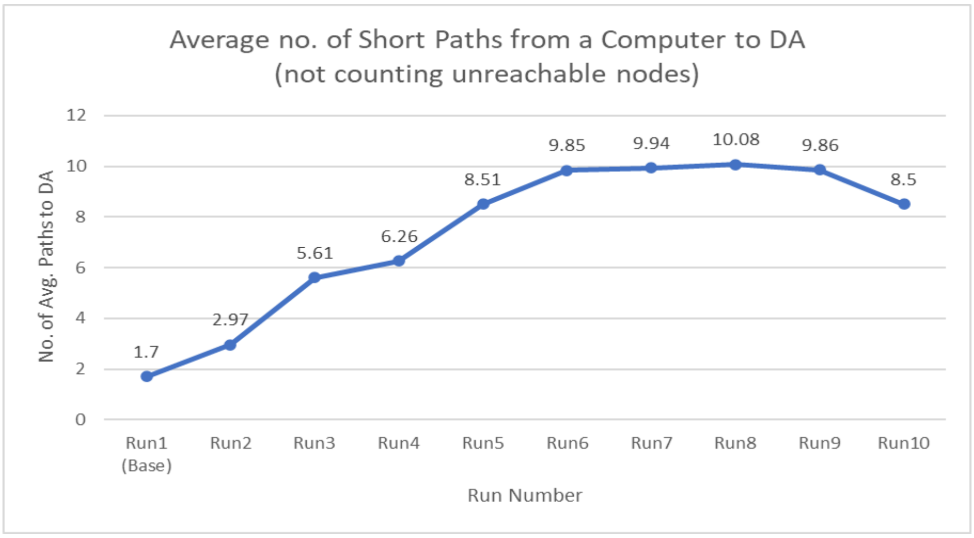


Figure 6: Avg. no. of steps from a Computer/User to DA
(based on a shortest path)

The figure above represents the average number of attack paths from a user/computer increases. Revealing old users/computers become an easy gateway for attackers to take control of the AD graph. Since many connections (to and from) are still active from the old usage and have never been cleared.

Finally, we can conclude that the complexity of the graph increases with each evolution cycle. Without any proper cleanup strategies, even a simple initial AD graph can grow into any IT administrator's nightmare. It also demonstrates the ripple consequences of a simple AD operation.

References

[1] Will Schroeder Andy Robbins, Rohan Vazarkar. Bloodhound database creator. https://github.com/BloodHoundAD/BloodHound-Tools/tree/master/DBCreator. Accessed: 2021-10-14.

[2] Will Schroeder Andy Robbins, Rohan Vazarkar. Bloodhound: Six degrees of domain admin. https://bloodhound.readthedocs.io/en/latest/index.html. Accessed: 2021-10-14.

[3] Raj Badhwar. *Advanced Active Directory Attacks and Prevention*, pages 131–144. Springer International Publishing, Cham, 2021. doi:10.1007/978-3-030-75354-2_13.

[4] Alan Burchill. Active directory structure guidelines - part 1. https://www.grouppolicy.biz/2010/07/best-practice-active-directory-structure-guidelines-part-1/, 2010. Accessed Feb. 02, 2022.

[5] Alan Burchill. Active directory structure guidelines - part 2. https://www.grouppolicy.biz/2010/07/best-practice-group-policy-design-guidelines-part-2/, 2010. Accessed Feb. 02, 2022.

[6] Alicex Johndunagan, Zheng, and Simon. Heat-ray:combating identity snowball attacks using machinelearning, combinatorial optimization and attack graphs. In *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles - SOSP '09*, page 305, Big Sky, Montana, USA, 2009. ACM Press. doi:10.1145/1629575.1629605.

[7] Alexander D. Kent, Lorie M. Liebrock, and Joshua C. Neil. Authentication graphs: Analyzing user behavior within an enterprise network. *Computers Security*, 48:150–166, 2015. URL: https://www.sciencedirect.com/science/article/pii/S0167404814001321, doi:https://doi.org/10.1016/j.cose.2014.09.001.

[8] S. Muthuraj, M. Sethumadhavan, P. P. Amritha, and R. Santhya. Detection and prevention of attacks on active directory using siem. In Tomonobu Senjyu, Parikshit N. Mahalle, Thinagaran Perumal, and Amit Joshi, editors, *Information and Communication Technology for Intelligent Systems*, pages 533–541, Singapore, 2021. Springer Singapore.

[9] Emilie Purvine, John R. Johnson, and Chaomei Lo. A graph-based impact metric for mitigating lateral movement cyber attacks. In *Proceedings of the 2016 ACM Workshop on Automated Decision Making for Active Cyber Defense*, SafeConfig '16, page 45–52, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2994475.2994476.

[10] Colin Tankard. Taking the management pain out of active directory. *Network Security*, 2012(4):8–11, 2012. URL: https://www.sciencedirect.com/science/article/pii/S1353485812700259, doi:https://doi.org/10.1016/S1353-4858(12)70025-9.

[11] AD Project team. Active directory at the university of bath. https://people.bath.ac.uk/ccsrsj/work/ad/. Accessed Feb. 02, 2022.

[12] Shannon Williams. Businesses under threat as attackers target active directory. https://itbrief.com.au/story/businesses-under-threat-as-attackers-target-active-directory. Accessed: 2021-10-14.