Department of Informatics
Technical University of Munich

TECHNICAL UNIVERSITY OF MUNICH

DEPARTMENT OF INFORMATICS

MASTER'S THESIS IN INFORMATICS

Exposing Insecure Configurations of Network Session and Permission Graphs

Thomas Maier

Technical University of Munich

Department of Informatics

Master's Thesis in Informatics

Exposing Insecure Configurations of Network Session and Permission Graphs

Offenlegung Unsicherer Konfigurationen von Sitzungs- und Berechtigungs-Graphen

Author:	Thomas Maier
Supervisor:	Prof. Dr.-Ing. Georg Carle
Advisors:	Simon Bauer,
	Dr. rer. nat. Holger Kinkelin,
	Thomas Penteker (Siemens AG)
Date:	May 15, 2019

I confirm that this Master's Thesis is my own work and I have documented all sources and material used.

Garching, May 15, 2019	
Location, Date	Signature

Abstract

In the past, several computer network attacks focused on obtaining credentials to proceed compromising. Furthermore, attackers made use of permissions of the stolen identity. Hereby, it is possible to even use indirectly assigned permissions by means of group memberships. This kind of lateral movement attacks are called Identity Snowball Attacks. This thesis tackles this specific problem in networks based on Microsoft Active Directory (AD). Moreover, the presented solution may be applied for a more general problem, i.e. for networks where administrative permissions are maintained similarly to AD networks.

Other approaches like the security-aware set-up and maintenance of AD networks are hardly implementable in large organizations as this solution may be rather time consuming. Even host-based approaches prove to be insufficient solutions. Hence, this thesis asks how to find undesired configurations in Network Session and Permission Graphs so that countermeasures can be initiated. The major goal of the introduced solution is to improve network security, especially by decreasing the risks of Identity Snowball Attacks.

This thesis describes typical attacker's behaviors regarding Identity Snowball Attacks. Hereby, we design configurations that potentially may be undesired. Those configurations consist of sub-structures within the graph that may be beneficial for an attacker and therefore support conducting Identity Snowball Attacks. It is shown, how to detect those configurations with graph-theoretic metrics.

Compared to other graph-theoretical approaches, the introduced solution is able to automatically analyze Network Session and Permission graphs with multiple centrality metrics: Degree, Betweenness and Closeness Centrality. The solution is evaluated with respect to its effectiveness by using randomly generated graphs. Thereby, undesired configurations are injected into the graphs. Afterwards, it is tested how well the configurations can be found by using centrality metrics. Finally, the functionality is tested with a graph originating from a significantly larger real AD network.

It turns out, that the solution enables an organization to successfully discover potentially undesired configurations by means of Degree Centrality and Betweenness Centrality. In contrast, the detection was not possible with the Closeness Centrality. Incidentally, we observe that another centrality metric – the PageRank – may be a comprehensive metric since the results strongly correlate with Degree and Betweenness Centrality. The PageRank metric seems to be the most effective centrality metric and therefore further research is necessary.

Zusammenfassung

In der Vergangenheit konzentrierten sich einige Angriffe in Computer-Netzwerken darauf, Zugangsdaten für eine weitere Kompromittierung zu bekommen. Zudem nutzten Angreifer die zugewiesenen Berechtigungen der gestohlenen Identität, wozu auch durch Gruppen-Mitgliedschaften indirekt erlangte Berechtigungen gehören. Diese Art von Lateral-Movement-Angriffen werden Identity Snowball Attacks genannt. Diese Arbeit widmet sich diesem speziellen Problem bezogen auf Netzwerke, die auf Microsoft Active Directory (AD) basieren. Außerdem kann die vorgestellte Lösung auch auf das generellere Problem – Netzwerke, die administrative Berechtigungen ähnlich zu AD verwalten – angewendet werden.

Andere Ansätze wie das sichere Aufbauen und Verwalten von AD-Netzwerken sind in großen Organisationen kaum umsetzbar, da diese Lösung zeitaufwendig sein kann. Es zeigt sich, dass auch Host-basierte Lösungen unzureichend sind. Daher stellt sich in dieser Arbeit die Frage, wie unerwünschte Konfigurationen in Sitzungs- und Berechtigungs-Graphen gefunden werden können, um Gegenmaßnahmen einzuleiten. Das Hauptziel der vorgestellten Lösung ist die Verbesserung der Netzwerksicherheit und vor allem das Verringern der Risiken von Identity Snowball Attacks.

Diese Arbeit beschreibt typische Verhaltensweisen von Angreifern bei Identity Snowball Attacks. Dabei werden Konfigurationen konstruiert, die potenziell ungewollt sind. Diese Konfigurationen bestehen aus Substrukturen innerhalb des Graphen, die vorteilhaft für den Angreifer sein und dadurch die Ausführung von Identity Snowball Attacks unterstützen können. Es wird dargelegt, wie diese Konfigurationen mit graphentheoretischen Metriken erkannt werden.

Anders als andere graphentheoretische Ansätze ist die vorgestellte Lösung fähig, Sitzungs- und Berechtigungs-Graphen mit mehreren Zentralitätsmetriken zu analysieren: Degree Centrality, Betweenness Centrality und Closeness Centrality. Die Effektivität der Lösung wird mithilfe von zufällig generierten Graphen evaluiert. Dabei werden ungewollte Konfigurationen in diese Graphen eingefügt. Es wird getestet, wie gut mit Zentralitätsmetriken vorher eingefügte, ungewollte Konfigurationen im Graphen wieder gefunden werden können. Schlussendlich wird die Funktionalität mit einem Graphen überprüft, der aus einem erheblich größerem realen AD-Netzwerk stammt.

Es stellt sich heraus, dass die Lösung Organisationen ermöglicht, erfolgreich potenziell ungewollte Konfigurationen mithilfe von Degree Centrality und Betweenness Centrality zu entdecken. Im Gegensatz dazu war eine Erkennung mithilfe der Degree Centrality nicht möglich. Nebenbei wurde beobachtet, dass die PageRank-Metrik übergreifend funktionieren könnte, da starke Korrelationen mit Degree und Betweenness Centrality erkennbar sind. Weitere Forschung ist notwendig, da es scheint, als wäre PageRank die effektivste Zentralitätsmetrik.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In 2015 John Lambert described today's challenges of defensive security teams and thereby introduced an alternative perspective on network attacks [1]. The attacker's point of view is different from the defender's perspective because the attacker pays attention to relationships between assets while the defender deals with lists of assets usually. Lambert paraphrases this imbalance with the title of a blog post:

*"Defenders think in lists. Attackers think in graphs.
As long as this is true, attackers win."*

Typical defensive activities in organizations often consist of limited protection techniques [1]: Signature-based malware scanning, key-word based detection of phishing emails, blocking IPs that belong to command and control networks and so on. Those are protection strategies based on lists.

In the last years, several attacks point out that a deeper understanding of the attackers' behavior is necessary to deal with the risk of becoming a victim of such attacks. Attacks have shown how offenders typically act, along with ransomware like BadRabbit or NotPetya or the well-known DigiNotar hack [2, 3, 4]. Attackers try to further move through the network after getting initial access to particular machines. All three above mentioned examples of the last few years have shown this behavior and all of them show traces to a tool called Mimikatz [5]. This software provides features that enable the attacker to steal user credentials on an already compromised machine, i.e. they may use them on other machines for further attacks in the network. Hence, the attacker uses the permissions of a user to access another machine where the attacker may gain more permissions via other users and so on. This quickly escalates as a kind of snowball system and ends in a large amount of compromised machines. Hence, such attacks are called *Identity Snowball Attacks*.

1.1 Motivation

It must be a major goal to make it as difficult as possible for an attacker to use existing sessions and permissions within a network for malicious purposes. There are several points to reduce the attacker's opportunities but it may be hard to implement solutions. Stealing credentials may be avoided with host-based solutions but there are still problems regarding the implementation within large organizations. Another approach may be to systematically plan user and group permissions. In large organizations this approach may end in high expenses due to complex organizational structures and a high number of interested stakeholders. Detecting such attacks with Intrusion Detection Systems (IDSs) or Intrusion Prevention Systems (IPSs) may be difficult as well as normal user traffic is hardly distinguishable from the attacker's traffic.

As a result, large organizations do not have a sufficient solution to limit the risk of the described attacks. However, despite such organizations have a focus on Microsoft Active Directory (AD), the problem scope may be broader than such specific use cases. This is why this thesis abstracts the specific problem of lateral movement within AD networks to a more general problem to solve.

1.2 Research Question

There is little research about the analysis of network session and permission graphs. The discovered related work have several disadvantages. Either high manual workload is necessary or only particular graph metrics have been considered. This is why this work builds a bridge between graph theory and computer networks regarding permission structures. Therefore the treated research question is as follows:

How to find Undesired Configurations
in Network Session and Permission Graphs?

The hypothesis resulting therefrom is that it is possible to discover configurations that may be beneficial regarding the attacker's behavior. This means, we assume that attackers show typical behaviors, that may be supported by particular topologies and permission structures – so-called configurations – within the graph. Such potentially undesired configurations are designed within this thesis by considering how they may assist the attacker.

There are existing solutions to collect information about established sessions and existing permissions within a network. We use this information in order to do an analysis from a central perspective. This is how components get detected within the graph, that may be interesting for the analysis.

Answering the above mentioned research question may lead to an improvement of the network security. The major goal is to narrow down opportunities for lateral movement by contributing a solution based on graph-theoretic metrics.

1.3 Outline

This thesis is structured as follows. At the beginning, the **Background (2)** chapter provides necessary foundations to understand the subsequent chapters. The next chapter introduces **Related Work (3)** for later comparisons to the designed solution. The **Analysis (4)** chapter analyses the scope of this thesis and examines various aspects of the underlying problem. After that, the **Design and Implementation (5)** chapter describes the solution design and realization, the **Evaluation (6)** chapter assesses how well the solution operates in different environments.

Chapter 2

Background

This chapter presents background knowledge that is used within the thesis. Hence, topics are covered that are strongly located in computer science like the possible ways of authentication and attacks on computer networks. Second, other fields contain necessary knowledge, like the graph-theoretical perspective on networks.

All along the history of computer networks, there is an accumulation of different possible ways to represent computer networks in order to emphasize particular perspectives on connected computers. This may be the perspective regarding the connections themselves as well as regarding attacks by utilizing those connections.

2.1 Authentication Mechanisms

Before taking a look at particular network connections, it is necessary to have a glance at basic authentication mechanism concepts. This is necessary in order to understand the graph-theoretical depiction of permission structures afterwards.

There are several different methods of authentication. Today's computer networks require authentication schemes in order to control who is able to access, modify and delete which resources. All of them have in common that systems are in need of some sort of credentials in order to authenticate a user. However, these schemes can be divided into the categories of centralized authentication databases and decentralized databases.

First, the centralized approach means that there is exactly one particular database that contains the credentials. This database needs to be queried in order to authenticate an individual user. Second, there is a decentralized strategy. This approach means that there are multiple locations to do authentications. That means that a user's credentials need to be placed in order to access specific resources. A user is required to authenticate against a database that maintains the access to those resources. If the access is granted, the user may use that specific resource. Here, the disadvantage is that a user is required to authenticate multiple times in order to access a particular resource.

There are hybrid forms of the centralized and of the decentralized authentication scheme as well. Examples for them are Open ID Connect (OIDC) or Kerberos which operate in a similar manner as similar strategies have been implemented. One example for those strategies is the *Single-Sign-On* concept. This means that a user is in need of an initial authentication against a fix authority. Afterwards, the user gets a successive secret in the form of a ticket in Kerberos or some token in OIDC for instance. In OIDC the responsive authority is called the OIDC Provider [6]. Kerberos uses the term Kerberos Domain Controller as an authority to authenticate [7]. The thereby gotten secret can be used for further authentication within the network. The user may use this secret to authenticate against other services to access further resources.

The advantage of this hybrid form is that the services are allowed to focus on the resource authorization. Hence, they are not responsible for authentication sequences anymore. The downside of such systems is that it may be enough for an attacker to steal the issued secrets. Afterwards the attacker may use this secret to act as the affected user. The attacker may use all permissions with the achieved secret that originally have been granted to the user.

2.2 Lateral Movement in Computer Networks

The drawbacks of the previously explained hybrid authentication may enable an attacker to move forward within a network. This is an important step for an attacker since there is a technique called Lateral Movement. It is a common attack technique and part of the typical phases of a computer network attack [8]. There are helpful tools like Mimikatz, that support an attacker in stealing user's credentials. In the particular case of this tool, the attacker can use it to extract a Kerberos ticket for instance. The attacker then may use this ticket to access resources in the name of the victim. This may also include the administrative access to further machines as well.

The MITRE corporation introduced the Framework to describe Adversarial Tactics, Techniques and Common Knowledge after compromise (ATT&CK). This framework states that Lateral Movement can be seen as the attacker's activities in order to move through the attacked network [8]. There are several counter measures against Lateral Movement. On the one hand, there are protection mechanisms on the network level to split networks from each other in order to prevent attackers from crossing it. On the other hand there are software tools that assist in the detection of attackers. Another protection mechanism is to comply with the least privilege principle [9] in order to lower the attacker's opportunities regarding the exploitation of permissions, e.g. administrator rights.

2.3 Active Directory

Microsoft developed AD as a collection of directory services to support an integrated environment [10]. This environment allows organizations to maintain a virtual depiction of itself. The directory services enable the organization to store and access information like user accounts, groups and computers. Those data objects can be used in the daily business and to ensure the technical operation of the network including the servers and workstations.

One use case may be to grant administrative permissions of a user or group to a particular computer or to establish membership relationships regarding AD groups. For instance, the `DOMAIN ADMINS` group can be used to maintain a set of users that allows them to administrate the computers within the organization [11].

A central component of an AD network – of a so-called AD *domain* – is the Active Directory Domain Controller (DC). There may be one or more DCs in form of Windows servers that enable administrators to maintain the centrally stored objects within the domain. Such objects are stored hierarchically and can be queried via network interfaces, like the Lightweight Directory Access Protocol (LDAP) interface. LDAP is a standardized protocol that allows to fetch that hierarchical information with the help of filter queries [12, 13].

Another relevant principle within this thesis is the so-called Group Policy Object (GPO). The GPO concept is used within AD domains to ensure the deployment of particular configurations within a domain [14]. The deployment can be done on a regular basis in order to distribute current settings onto all computers within the AD.

2.4 Information Gathering in Windows Domains

Major dependencies regarding this thesis are possible scanning techniques in Windows networks. Typically, an attacker scans networks in order to collect information about the network, its users, computers, and permissions. In the case of this thesis we need to explain scanning opportunities.

This gathering of information may be used out of a defensive perspective as meaning scanning of the Windows network from a defender's point of view. The advantage of this option is that more information may be found out about the network, since administrative access can be used for collecting information as well. The disadvantage of this perspective is not that realistic compared to a real-world attack. Therefore it may be interesting to scan the network in a black box manner, i.e. imitating the attacker's behavior. This approach may allow a more reasonable analysis regarding real-world attacks, despite this means a smaller data set.

The following sections describe scanning techniques that are relevant concerning this thesis.

2.4.1 Local Admin Enumeration

In Microsoft Windows every particular computer has its own local administrator group, called `Administrators`. The members of this group have administrative access to the local machine.

Microsoft Windows provides function calls based on the Windows API (formerly called Win32 API). More specifically, the NETAPI32 provides the `NetLocalGroupGetMembers` function to query group members of remote hosts [15]. The mentioned function consists of multiple low-level remote calls via the `MS-SAMR` protocol, that has been specified by Microsoft [16].

Particular Windows API calls require different authentication sequences. For the case of the mentioned `NetLocalGroupGetMembers` function it is possible to call it anonymously, i.e. without any authentication. However, it is possible to restrict access to authenticated users. This preference has to be set regarding the desired remote host in the form of registry keys [17]. This can be done by editing registry keys directly on a particular host or via the setting of GPO. There are several *Security Options* to edit in order to restrict anonymous access within the entire Windows domain [18].

It is necessary to reach out to each and every computer to collect data for all described options. Alternatively it is possible to scan in a stealth manner to collect information about local admin groups [19]. To do this, information can be collected by querying the DC only. This can only be achieved if the DC maintains local admin groups through the entire domain by making use of GPOs. This might be slightly inaccurate, since GPOs may get changed locally. Those changes cannot be determined because the DC does not detect changes necessarily. Despite of this, GPOs may get deployed on a regular basis and therefore overwrite changes.

2.4.2 Session Enumeration

Data about established remote sessions are part of the data set used within this thesis as well. This data can be collected in two different modes [19].

One possible way to gather session data is a Windows API function called `NetSessionEnum` [20]. By invoking this function it is possible to fetch information about currently established sessions on the remote host. It provides the source of the connection to the host in the form of an IP address and the corresponding user that is logged in. This API call is possible without any authentication mechanism [19].

The second way to collect information about established sessions is a similar Windows API function called `NetWkstaUserEnum` [21]. This function call is more comfortable than `NetSessionEnum` because it provides more information about the host and about the user, e.g. the authenticating domain [19]. However, this invocation requires administrative privileges on the requested target hosts. Therefore using this function makes sense only out of a defensive perspective or if the attacker achieved administrative permissions already. Furthermore, having administrator's rights opens up the possibility of using the registry of the remote host to collect further information.

2.4.3 Group Membership Collection

AD provides various service interfaces, among them there is also an LDAP interface. LDAP queries are possible via this interface by using filters as specified with the LDAP versions 2 and 3 (see RFC1777 and RFC3777) [22, 12, 13]. This interface may be used for the collection of users, groups and computers that are stored as part of the AD domain on a DC [19].

2.5 Centrality Metrics in Graph Theory

The sections before this one explained the practical background regarding Windows networks. This part addresses a more theoretical piece within this work. As previously mentioned, virtual relationships between users, groups and computers in an AD domain can be collected (see Section 2.4 for further description). Those relationships may be depicted with as a connected graph. This thesis is about the analysis of such graphs. Therefore this section explains graph-theoretical concepts with respect to the analysis of graph components.

In graph theory *centrality metrics* have been introduced to identify vertices with a higher influence regarding a graph [23]. A large number of different algorithms is in use for various applications. Originally centrality metrics popped up in the area of social networks. Thereby, those algorithms have been used in order to find persons that have a higher social influence regarding their environment. Each metric is designed in order to discover "central" vertices in different senses. Within this thesis, a few centrality metrics are used and therefore the background needs to get explained within this chapter. This section sums up introductory information about centrality metrics using an article called "Centrality and Hubs" edited by Fornito et al. [23].

2.5.1 Degree Centrality

The first centrality metric presented is the Degree Centrality. It is defined as the amount of neighbors and called *node degree* or *vertex degree* as well. Thus, the Degree Centrality DC of a vertex i is defined as follows:

$$\begin{aligned} DC(i) &= \deg(i) \\ &= \sum_{i \neq j} A_{ij} \end{aligned} \tag{2.1}$$

$\deg(i)$ may be calculated with the help of an adjacency matrix A. The entries A_{ij} in this matrix 2.1 depict a connection between vertex i and vertex j. If there is a direct edge between the vertices, the value is 1, otherwise it is set to 0. Therefore the sum calculates the amount of connections from vertex i to other vertices. The higher the DC of a vertex gets, the more direct neighbors are present.

2.5.2 Betweenness Centrality

The Betweenness Centrality score calculates the amount of shortest paths a vertex lies on. The equation 2.2 shows how this metric operates.

$$BC(i) = \frac{1}{(N-1)(N-2)} \sum_{h \neq i, h \neq j, j \neq i} \frac{\rho_{hj}(i)}{\rho_{hj}} \tag{2.2}$$

$\rho_{hj}(i)$ is the amount of shortest paths between vertices h and j that cross vertex i. Below ρ_{hj} is the number of shortest paths between h and j. Therefore, the ratio between shortest paths that include i and the total amount of shortest paths gets calculated. This value gets divided $(N-1)(N-2)$ which is the amount of vertex pairs without vertex i.

Therefore the Betweenness Centrality score is a metric that shows how deep a vertex is connected within the graph regarding the amount of shortest paths the vertex lies on.

2.5.3 Closeness Centrality

The Closeness Centrality may be described as a metric that shows how close a vertex is to all other vertices within the graph. Therefore this metric is the inverse of the average amount of steps on the shortest path to all vertices.

Equation 2.3 defines the normalized Closeness Centrality for a vector i.

$$CC(i) = \frac{N-1}{\sum_{j \neq i} l_{ij}} \tag{2.3}$$

Thus, CC is calculated by taking the total amount of vertices N without vertex i. All shortest path lengths l_{ij} between vertex i and j are summed up in order to calculate the average closeness to all other vertices within the graph.

Chapter 3

Related Work

This chapter introduces two approaches that have been developed in the past by other organizations. The Heat-ray project [24] as well as the BloodHound project [25] designed different approaches that enable an organization to discover undesired configurations within an AD network. The following sections explain their strategies in detail. A comparison to our solution can be found in Section 6.5.

3.1 Heat-ray

Dunagan et al. developed a system called Heat-ray to combat attacker's Lateral Movement within AD networks [24]. Within this context they introduced the term *Identity Snowball Attack* as well. This section explains the underlying concept they developed.

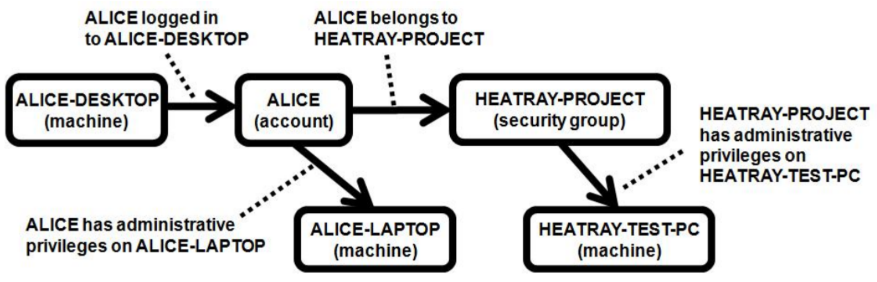
ALICE logged in to ALICE-DESKTOP
ALICE belongs to HEATRAY-PROJECT
ALICE-DESKTOP (machine)
ALICE (account)
HEATRAY-PROJECT (security group)
HEATRAY-PROJECT has administrative privileges on HEATRAY-TEST-PC
ALICE has administrative privileges on ALICE-LAPTOP
ALICE-LAPTOP (machine)
HEATRAY-TEST-PC (machine)

Figure 3.1: Example for a network session and permission graph [24]

The system developed by Dunagan et al. is based on machine learning techniques and combinatorial optimization. Specifically they make use of attack graphs as can be seen in Figure 3.1.

3.1.1 Identity Snowball Attacks

The term "Lateral Movement" is defined as follows for the remainder of this thesis, unless explicitly stated otherwise. This thesis is about the traversal of permission structures, meaning that the attacker is assumed wandering through the permission graph in a virtual manner. This kind of Lateral Movement is called *Identity Snowball Attack* in the context of the Heat-ray project [24, 26]. Those virtual movements still may end in the compromise of further computers within the network. This is because the attacker may exploit the permission structures as explained in 3.2.1.

Figure 3.1 shows an example for a permission graph as used by Dunagan et al. [24]. If the attacker compromised the computer *Alice-Desktop*, the attacker may use every permission of Alice since Alice is logged in on that machine. Thus the attacker is able to use Alice' identity to log in onto *Alice-Laptop* and *Heatray-Test-PC* as well.

3.1.2 Iterative Heat-ray Process

Dunagan et al. defined the following iterative process. This section explains the meaning of the particular steps.

1. Execution of sparsest cut algorithm
2. Group and rank edges
3. Assessment by IT administrators
4. Execution of a Machine Learning algorithm to deduce edge costs

In Step 1 the sparsest cut algorithm is executed on the existing graph. This algorithm tries to find the smallest set of edges, whereby a removal of them results in two separate graph components. Both components shall be as large as possible. This set of edges may be found with a combinatorial optimization technique as defined by the Heat-ray authors. Basically, two different values take effect on the chosen set of edges – the edges' benefit and the cost of removing it.

The benefit is derived from the amount of shortest paths that pass the edge, meaning that the benefit of cutting this edge is higher. This is because the probability of a passing attacker (during the lateral movement) may be higher for those nodes. Therefore, it makes sense to cut those edges. The problem with this value is that the deletion of some edges may cause more problems than the deletion of other edges. This might be the case if the effort to delete an edge is lower than deleting another one. This is where the edge costs come into play. However, get changed at a later point in the Heat-ray process. The lower the edge costs are and the higher the benefit is, the better it would be to propose the deletion of an edge.

Step 2 has been inserted to support the system administrator in the next step. This step allows the system administrator to make less decisions that take effect on multiple nodes at the same time. The system administrator has to decide whether a group of edges should be cut or whether it should be kept (see Step 3).

This decision influences Step 4 where a Machine Learning algorithm uses the administrator's feedback to set edge costs through the entire graph. This is necessary since it would be too much effort to set the costs for all edges in a manual manner.

In summary, Heat-ray uses the feedback of an IT administrator and fitting edges in order to cut a network session and permission graph into components that are as large as possible. This ends in a more loosely connected graph, whereby an attacker has little opportunities to access further large graph components. This is how Heat-ray tries to decrease the amount of configurations within the graph that are beneficial to the attacker. At the same time Heat-ray tries to decrease the effort that an IT administrator has to achieve the before-mentioned cutting of the graph.

3.2 BloodHound

Vazarkar et al. introduced tools as part of a project called BloodHound AD [25, 27, 28]. The tool set consists of SharpHound to scan AD domains, the BloodHound user interface to analyze the data and DBCreator to randomly generate test data. In general, the goal of the BloodHound project is to provide software to reveal potentially insecure configurations within AD networks. The approach relies on the IT administrator's ability to manually check crucial paths with the help of BloodHound. Those configurations need to be checked by some security team in order to assess appropriate changes to the configuration.

3.2.1 Scanning with SharpHound

It is possible to gather data via several interfaces in Windows networks as explained in Section 2.4. Those interfaces are utilized by SharpHound. SharpHound is the latest implementation of the available ingestors, as the BloodHound AD developers call their data collectors. In general, ingestors collect information about the Windows network to insert it into the Neo4j database as nodes and relationships.

Thereby SharpHound creates different kinds of nodes, that are differentiated by using the following distinguishable node labels: `Computer`, `Group`, `User`. This means, that for every computer, group and user one node will be created. Those nodes will get connected with various relationships depending on the results of the scans explained in Section 2.4.

The following table explains the source and meaning of each relationship type. SharpHound persists each relationship as a directed edge in order to generate an entire graph of the AD network.

Type	Source	Meaning	Source node(s)	Sink node(s)
`has-session`	Local computer (session enumeration)	Currently established user sessions of a computer	Computer	User
`member-of`	DC	LDAP Group memberships of groups and users	User, Group	Group
`admin-to`	Local computer (group enumeration)	Local administrative rights	User, Group	Computer

Table 3.1: SharpHound relationship types

As can be seen in Table 3.1, the Win32 API of the local machines provide information about established sessions (`has-session`) and group memberships of the local admin group (`admin-to`). The DC is to be queried by SharpHound regarding domain group memberships (`member-of`) because such relationships are stored centrally within the AD.

3.2.2 BloodHound Analysis User Interface

Vazarkar et al. developed a tool called BloodHound that enables the analysis of SharpHound permission graphs [25]. Afterwards the BloodHound application may be used to analyze the existing graph. The BloodHound project provides the tool in the form of a graphical user interface for the analysis of the generated graph. It can be used for investigations regarding particular nodes or to search for node-driven or context-specific characteristics. Examples for those queries would be:

- *Find all AD domain administrators!*
- *Find the shortest path(s) between two specific nodes!*

BloodHound may be applied by offensive and defensive security teams. One possible case may be to map the network with SharpHound and then to use BloodHound to understand or analyze the permission structure within the AD [29]. Defensive security teams may be interested in the defensive perspective especially [30]. They are able to do risk auditing through the AD network with BloodHound. Viewing possible attack paths enables them to gain a better understanding of the attacker's perspective onto the network. Furthermore, those analysis techniques may be beneficial regarding the mitigation of discovered risks.

3.2.3 DBCreator

The BloodHound developers implemented a tool called `DBCreator` in order to test use cases of the BloodHound Analysis User Interface [28]. It consists of a script that dynamically creates graphs that depicts fictional Windows network environments. The tool may randomly create graphs that shall be as similar to a real-world AD environment as possible. The thereby generated data can be used as an alternative data source to test the BloodHound Analysis User Interface or to test other analysis platforms. The size of the generated database can be set as a parameter. Nodes and relationships between them are generated in the same way as explained in Section 3.2.1 for the SharpHound ingestor (see Table 3.1).

The generated graphs created by the `DBCreator` show up with similar structures because the developers tried to simulate real-world environments as accurate as possible. This ends in rather similar structures, e.g. security groups are generated for IT departments with a number of employees. Hence, the graphs have similarities from a graph-theoretic perspective. Thereby, it is not possible to generate totally random graphs, since these graphs are used to test the functionality of the BloodHound analysis tool that is built for real-world environments. This means that typical characteristics of organizations still show up in the random graphs, but the concrete structures are randomly chosen. This ends in graphs that still contain rather real-looking organization structures.

Chapter 4

Analysis

This chapter describes the fundamental problem treated within this thesis and several possible approaches to tackle the problem. At the end of the chapter, the requirements for our solution are defined. This is followed by an analysis of the attacker's behavior at the end of this chapter.

4.1 Problem Statement

The major objective of this thesis is to prevent Identity Snowball Attacks. This type of attack has been defined within the Heat-ray project as described in Section 3.1.1. This term depicts a special kind of Lateral Movement where the movement refers to the usage of a network session and permission graph.

Within this thesis the treated problem can be divided into a general problem and a specific use case. The general problem is the prevention of Identity Snowball Attacks regarding network session and permission graphs. The specific use case are Identity Snowball Attacks regarding AD infrastructures.

We assume that the specific use case can be generalized so that the later introduced solution can be applied to prevent Identity Snowball Attacks on all permission graphs.

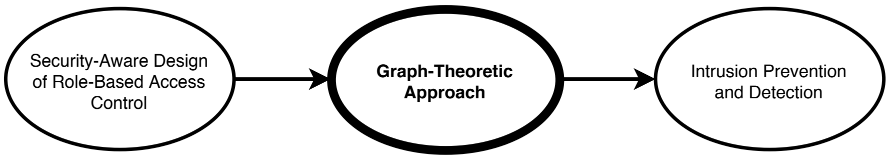
Security-Aware Design of Role-Based Access Control
Graph-Theoretic Approach
Intrusion Prevention and Detection

Figure 4.1: Temporal classification of our approach

The reason for this is that the used graph within this thesis can be generated for other specific use cases in a similar structure.

Figure 4.1 shows points in time where it may be possible to apply measures in order to prevent or detect Identity Snowball Attacks. The following sections give more detailed descriptions of the depicted points and possible countermeasures. Furthermore, the reasons for the final decision regarding the Graph-Theoretic Approach becomes more clear.

4.2 Security-Aware Design of Role-Based Access Control

The access to data within an organization is controlled with a permission system typically. For instance, such a system controls which users or groups may have administrative permissions on resources. Microsoft reflected this concept by using AD together with specific permission schemes. Within the concepts Microsoft made use of Role-Based Access Control (RBAC).

Regarding AD domains, this means that the permission data is located on various resources. Therefore this data is maintained in a decentralized manner – parts are located on DCs or even on workstations.

However, this data may be collected in order to persist it as a connected permission graph in a database. One opportunity to avoid Identity Snowball Attacks is to tackle the problem during the construction of the permission graph. The first point to take a look at is the initial planning phase of the permission graph. The second point in time to consider is to permanently alter the existing graph.

In order to avoid Identity Snowball Attacks it is necessary to pay attention on the built sub-structures. This becomes a very complex task in large organizations with high amounts of computers, groups and users. On the one hand, there are more than one authority that takes care of the administration of the entirety of the organizational permission patterns. This makes it difficult because of diverse requirements and lots of necessary communication. On the other hand, changes within the graph happen in a fast manner. Typically many activities happen in a short time like granting permissions, creation of new users or the modification of group memberships. A typical user or administrator is not able to access the whole permission structure because of organizational structures or security measures. These rapid changes out of a "local perspective" within the graph make a security-aware design of RBAC difficult.

4.3 Intrusion Prevention and Detection

This section refers to the third circle in Figure 4.1. First, there are tools to detect or prevent the exploitation of vulnerabilities in the form of host-based security measures. Furthermore there are network-based approaches to mitigate the problem of Identity Snowball Attacks with the aid of IDS/IPS.

4.3.1 Host-Based Security Measures

There are several security measures that can be done directly on the specific host within the network. Thus attackers need to take further steps in order to circumvent those measures. One possible solution regarding this thesis would be using the software *Windows Defender Credential Guard* [31]. This tool has been released by Microsoft in order to protect the system from *Pass the Hash* attacks [32]. The developers of the tool Mimikatz are able to steal credentials and reacted with additional features in order to bypass Microsoft's protecting tool [33].

There are still additional problems with using the *Windows Defender Credential Guard* on all Windows computers. It is necessary to set the *Hyper-V* flag on all machines in order to get *Windows Defender Credential Guard* working. This is possible only if no other virtualization technology (like VirtualBox or VMWare) needs to be used on that machine without continuously (un)setting the flag and rebooting the machine [34].

On the other hand there are restrictions concerning the requirements of the *Windows Defender Credential Guard* [35]. The software is deployable on Windows 10 and Windows Server 2016 only. Therefore it is currently impossible to use it within heterogeneous networks consisting of machines with other Windows versions installed.

Summarizing the above, it can be said that host-based measures are not always sufficient. They can be effective on a subset of all machines but they cannot ensure protection for the whole network.

4.3.2 Network-Based Security Measures

Identity Snowball Attacks exploit normal user permissions as explained in Section 3.1.1. Permissions that are already existing are used in order to access other computers within the network. This is the case not only for Active Directory, but for other systems that are built up on similar permission systems. Differentiating malign and benign network traffic is a complex task because there is a variety of different protocols that need to be supported. Furthermore it is difficult to identify an attacker's traffic if the attacker makes use of normal user API calls. This is why detecting attacks with network-based security measures is difficult.

The analysis of such high-level protocols gets even harder if the associated protocols are encrypted. Further problems arise if logging mechanisms are distributed through multiple machines within the network. This means that distinguishing user and attacker behavior requires a central evaluation of all activity logs.

4.4 Graph-Theoretic Approach

In summary, it can therefore be said that a security-aware design of the permission graph is hard to achieve in the described environment. The stated approaches regarding live detection/prevention cannot be deployed widely as well because of the heterogeneous nature and complexity of the network. This shows that the most profitable approach for a solution may be in order to analyze the permission graph.

This results in the advantage of a centralized approach. Thereby, the analysis is possible at a central point without the necessity of time-consuming communication with affected stakeholders. Another advantage is the possibility of using the scanning tools developed by the BloodHound project (see Section 3.2.1).

4.4.1 Investigation Process

Our solution relies on the previously described graph built by SharpHound (see Section 3.2.1). We make use of those graphs within a novel process that is to be described in this section.

IT security teams are in need of a process in order to secure permission graphs to avoid Identity Snowball Attacks, e.g. in the case of AD networks. We suggest to establish the following steps as an integrated process to solve this problem.

1. Scanning
2. Graph-Theoretic Analysis
3. Evaluation of Results

At Step 1 a scan needs to be executed in order to collect and persist the graph data as described in Section 3.2.1. Within Step 2 a graph-theoretic analysis of the existing graph needs to be invoked. The analysis results in the proposal of graph sub-structures that depict potentially undesired configurations. The security team needs to evaluate those proposals in Step 3. This is necessary in order to determine the next steps regarding

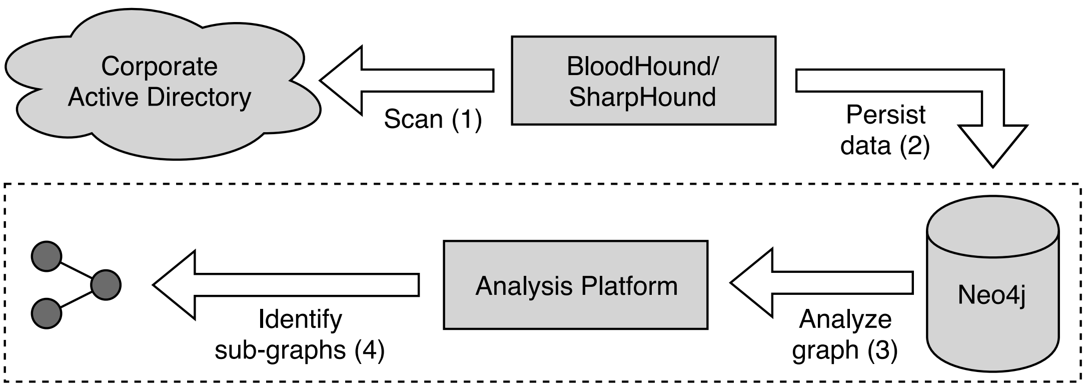

Figure 4.2: Technical perspective of the investigation process Steps 1 and 2

the particular sub-structures. Possible reactions may be the deletion or modification of a graph substructure.

The focus of this thesis is on Step 2 of the investigation process. Thereby concepts need to be developed in order to analyze the graph in a way that allows the proposal of potentially undesired substructures. Therefore we want to find substructures that may favor the behavior of an attacker. Learning about such substructures enables security teams to initiate subsequent actions to reduce the risk of Identity Snowball Attacks.

Figure 4.2 depicts the chronological sequence of Step 1 and Step 2 from a technical perspective. Thereby the process is applied onto an AD network. First, the AD networks needs to be scanned (1) in order to persist the data in a graph database (2). Afterwards the analysis platform searches the graph for undesired substructures (3) in order to propose them to the security team (4). The dotted box in the lower part of Figure 4.2 depicts the scope of this thesis according to Step 2 of the investigation process. Therefore, the focus is set on the last part of the depicted diagram in figure 4.2.

4.4.2 Requirements

The primary problem to solve within this thesis is to identify undesired configurations within permission graphs. Hence, this section describes the requirements for the wanted solution at first. Afterwards, we introduce undesired graph configurations by making use of centrality metrics.

At first, there are two more general non-functional requirements. The solution to be developed needs to be fast enough to use it in the iterative process described before in Section 4.4.1. This means that Step 2 needs to be fast enough to use the tool interactively. The security team should be able to start the graph-theoretic analysis within the process as a day-to-day habit. Hence, the resulting requirement is a fast

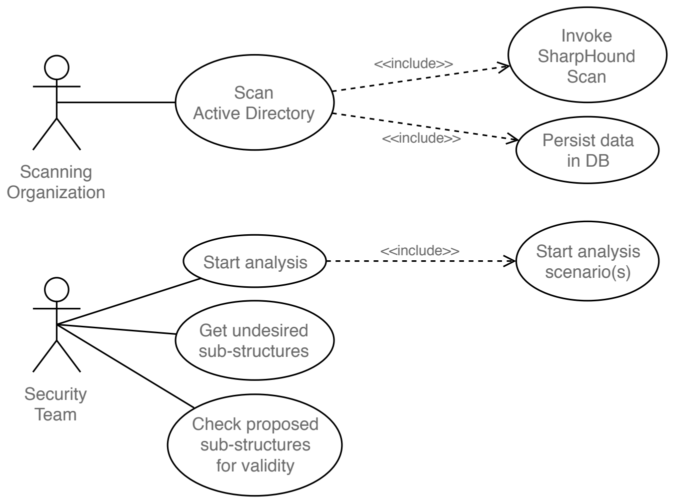

Figure 4.3: Use Case Diagram

execution time, so that the security team is able to do the analysis in a short amount of time. Afterwards the team is ready to start with the result evaluation in Step 3 of the investigation process. Therefore, the non-functional requirement is that the solution should be applicable for graphs collected in large organizations. This requires the software to efficiently process large amounts of data in a short time.

Another non-functional requirement is that the solution may be applied to general network session and permission graphs. As mentioned before, the specific problem regarding this thesis relates to AD domains. Hence, the collected and generated graphs within this thesis are biased with AD structures. However, each and every consideration is done with the background of general network session and permission graphs, but not AD networks only. Therefore, the hereby designed strategies including all algorithms can be applied to general graphs.

Figure 4.3 shows the stakeholders that take part in the process described in Section 4.4.1. This thesis puts emphasis on functionality that has to do with the analysis procedure especially. In the following sections we define the specific requirements of the wanted solution.

Sub-Structure Proposals

As mentioned before, we need to identify undesired graph substructures to the security team. We use the term undesired substructure in order to describe structures within the graph that favor the attacker's behavior. If an attacker benefits from the particular constellation within the graph. The main goal within this thesis is to define such constellations and to find a way to find them in a fully-automated manner.

Static Blacklisting

There are some cases in network session and permission graphs that require the exclusion of substructures from the analysis process. We identified two cases that enforced such exclusions.

First, there may be undesired substructures within the graph due to technical reasons. On the one hand, one may argue that such substructures shall be proposed because they are declared to be undesired. On the other hand, we need to avoid such proposals in the output of the analysis to avoid obvious results. One example for this kind of substructures to be blacklisted is the `DOMAIN ADMINS` group within an AD network (see Section 2.3). This node needs to be declared to maintain the group of users that have administrative permissions throughout the entire AD domain.

Second, there may be organizational reasons for particular substructures. For instance, it is possible to map organizational structures onto permission graphs. In the case of AD it may be appropriate to construct tree structures to construct pyramidally set up divisions or departments. Especially nodes that lie on top of such structures may seem like undesired configurations at a first glance. Security teams may want to exclude such nodes despite of that because these substructures are well-known already.

These two types of undesired substructures need to be blacklisted to exclude them during the analysis procedure.

Dynamic Blacklisting

There is an additional kind of blacklisting technique. Other than the static blacklisting method, it is necessary to exclude substructures in a more dynamic way. With the static blacklisting it is possible to exclude explicitly defined nodes for instance. The goal of the dynamic blacklisting is to exclude more complex substructures that need to be defined in an agile manner. Within an AD network it may be interesting to exclude whole groups with its members for instance. One example may be the exclusion of not only the `DOMAIN ADMINS` group but all members (therefore all admin users)

Randomly Generated Graphs

In order to evaluate whether the described requirement in 4.4.2 works well, we need to generate random graphs. Another reason is that during this thesis we do not have access to a large number of real-world graphs. The following chapters describe how random graphs are used to test our solution. Therefore those random graphs need to be as similar as possible in regards to real environments. Those graphs are necessary to check whether graph substructures can be found as expected.

Injection of Subgraphs

As described earlier the major objectives of this thesis is to find undesired substructures in graphs. Therefore, it is an advantage to synthetically construct undesired substructures. Those subgraphs can be seen as examples for undesired substructures that we want to find in real graphs.

There are two use cases for the injection of subgraphs in order to evaluate our approach and solution. First, undesired substructures need to be injected into randomly generated graphs (as described in Section 4.4.2). Second, the same substructures are injected into a real-world graph. This is how we test our solution in order to ensure that similar undesired substructures are detected by our approach.

4.4.3 Attacker's Behavior

This section states how different forms of graph configurations may have different effects on the attacker. We assume that undesired graph configurations favor the attacker's opportunities regarding Lateral Movement. Therefore we describe attacker behaviors in order to reflect attacker-friendly configurations.

The following sections describe assumed behaviors of attackers. Those behaviors are matched onto metrics in order to find undesired configurations within the graph. Out of a defensive perspective, those metrics are used to assess the risk of compromise of individual nodes. We present one example per metric with a figure where users are depicted as purple circles. Groups are depicted as blue and computers as orange circles.

The metrics defined in Section 2.5 may be used despite they are introduced as metrics for undirected graphs. The directions of the relationships are irrelevant since the attacker profits from each direction of the relationships.

Direct Neighbors of Particular Nodes

As explained in Section 2.5.1 the Degree Centrality represents the amount of direct neighbors of a node. Therefore there are two cases where this centrality metric seems interesting in comparison to the attacker's behavior.

- On the one hand a node is interesting for an attacker if the attacker has information about the environment of a node. The attacker profits the most of such a node if multiple nodes are wanted as the final target. First, it is possible to use a node with a high Degree Centrality to reach the mentioned targets initially. Second, the attacker may stay at this node in order to keep it as a profitable starting point for future activities.

- On the other hand, an attacker who does not have any information about the network may do a random walk through the graph. Therefore the probability of the attacker reaching a node with a high Degree Centrality is more likely. This applies particularly from a local perspective since the direct neighbors have lower scores. Additionally it is more likely that an attacker compromises more nodes directly after a nodes with a high Degree Centrality since there are more direct neighbors.

Those two cases indicate that nodes with a higher Degree Centrality score are more likely to be undesired substructures than nodes with lower scores.

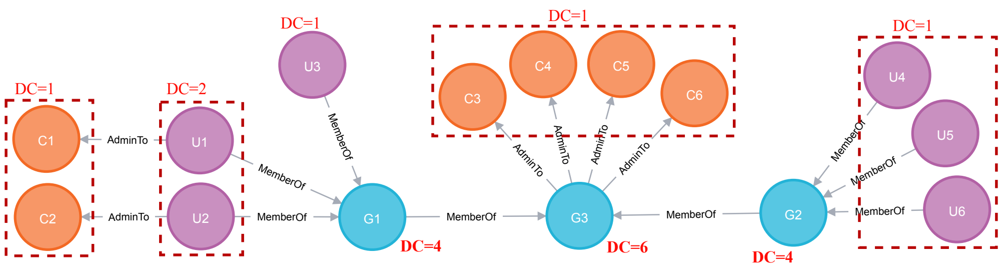

Figure 4.4: Example graph with calculated Degree Centrality (DC) per node

Figure 4.4 shows an example graph with the associated Degree Centrality scores (DC). This and all following figures of this type shows users as purple, groups as blue and computers as orange circles. The red dotted boxes frame nodes that obtain the same scores.

Out of a local perspective it can be deduced that stealing the identity of User `U1` is more favorable for the attacker than the user `U3` for instance. This can be ascribed to the fact that `U1` has administrative permissions on computer `C1` and is a member of group `G1` at the same time. Therefore the attacker's profit is higher in contrast to `U3`.

Out of a global perspective it is more likely that an attacker passes `U1` than `U3` by doing a random walk. This is because there are more paths where the attacker may use the identity of `U1`. Furthermore the attacker is able to use more paths in order to compromise more nodes afterwards.

Thus Degree Centrality may be used in order to find potentially undesired nodes within the graph.

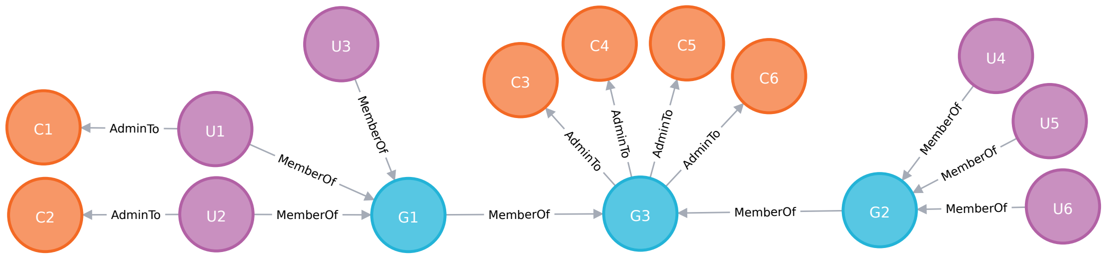

Figure 4.5: Example graph with calculated Betweenness Centrality (BC) per node

Nodes on the Attacker's Path

We assume that nodes have a higher probability of compromise if they lie on many shortest paths within the graph. The affected algorithm to measure this metric is called Betweenness Centrality as explained in Section 2.5.2. Dunagan et al. use the same metric with a similar reasoning (see 3.1).

We argue that nodes with higher Betweenness Centrality scores have a higher chance of getting compromised because of two reasons.

- In graph-theory a higher Betweenness Centrality means that the node lies on many shortest paths between two nodes within the graph. Regarding an attacker this means that the probability of passing such a node is higher than passing a node with a lower score. This reasoning works, regardless whether the attacker does a random walk or with a specific target. The higher the score, the higher the probability of a node getting compromised.
- There is still another perspective regarding a higher score of a node. A high Betweenness Centrality implies that there are many possible shortest paths to the node itself. Therefore the risk regarding such a node is higher because an attacker has more opportunities to reach that particular node.

In Figure 4.5 one may recognize that group `G3` is in possession of the highest Betweenness Centrality (BC) score. This is because this node separates most of the nodes and therefore lies on the shortest path between them. Group `G1` is affected in a greater extent in comparison to group `G2` because `G1` splits a higher amount of nodes within the graph as can be seen.

Taking a look at the computers `C3`, `C4`, `C5` and `C6` show a contrasting situation. They do not lie on any shortest paths of two other nodes within the graph and therefore do have a score of `0`. This does not mean that no risk stems from those nodes, but the Betweenness Centrality metric brings no advantages regarding them.

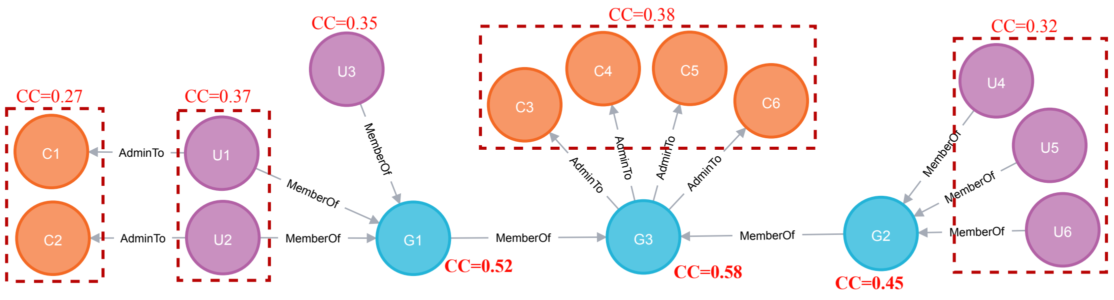

Figure 4.6: Example graph with calculated Closeness Centrality (CC) per node

Closeness to the Attacker

The last example in Section 4.4.3 states an example where Betweenness Centrality is an inappropriate metric to find the mentioned computers. Figure 4.5 shows that the score of C3, C4, C5 and C6 is very low. Still, we assume that those nodes have a greater risk of getting compromised than other structures within the graph for the following reason: Those nodes do not lie directly on paths that may be used by an attacker probably. Even though we assume them being at risk because group C3 is a direct neighbor with a high Betweenness Centrality apparently. This case may be covered with an additional centrality metric called Closeness Centrality.

In Figure 4.6 can be seen that the Closeness Centrality (CC) score of C3, C4, C5 and C6 is higher than the score of other leaf nodes like C1 or C2. Even stronger integrated nodes like users U1 or U2 obtain lower scores because the discussed nodes are integrated in a stronger manner and therefore they are closer to other nodes.

There are two perspectives to be noted again regarding the risk of compromise.

- First, a node is more likely to get compromised if a node lies close to other nodes that are compromised already. Therefore a higher Closeness Centrality score means that a randomly walking attacker is more likely to reach that node on the future way through the graph. Hence, the node is at greater risk regarding soon-to-be compromising.
- Second, a node with a higher score presents greater risk if it is compromised already. The increased Closeness Centrality score is to be understood as a higher risk for other non-compromised nodes since the compromised one is closer at an average.

Thus, a higher Closeness Centrality score is seen as an indicator for nodes that represent higher risk than nodes with lower scores.

Chapter 5

Design and Implementation

This chapter provides an overview of how this thesis tackles the previously analyzed problem of finding undesired configurations.

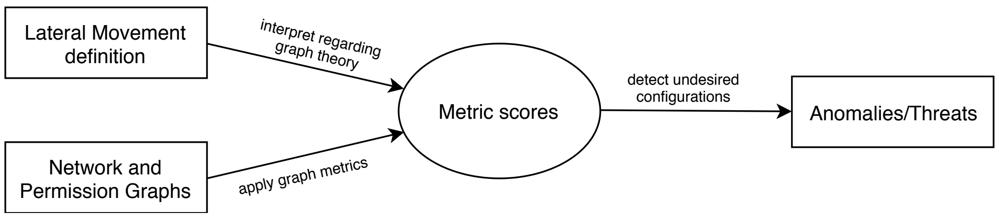
Lateral Movement definition
interpret regarding graph theory
Network and Permission Graphs
apply graph metrics
Metric scores
detect undesired configurations
Anomalies/Threats

Figure 5.1: Methodology of this thesis

Figure 5.1 shows how the following sections combine the previously explained concepts. On the one hand, the previous chapters explain the definition of lateral movement can be put in the context of network session and permission graphs. On the other hand, Chapter 4 describes how graph analysis may be used to expose anomalies within the graph. Figure 5.1 visualizes those concepts to explain the terminology. Graph metrics need to get applied onto the graph to interpret the scores in order to discover undesired configurations. Within the scope of lateral movement, those undesired configurations may be beneficial regarding the attacker's behavior. Hence, this strategy may expose anomalies and therefore threats to particular sub-structures of the graph.

This chapter describes how the developed solution makes use of the previously analyzed concepts. Subsequently, the designed architecture is described and thus bridges the gap to the applied methodology.

5.1 Analysis Platform Components

This section presents an overview of the concept of our solution. There are several steps necessary in order to propose undesired sub-graph structures. In the following we describe the individual modules within the analysis platform.

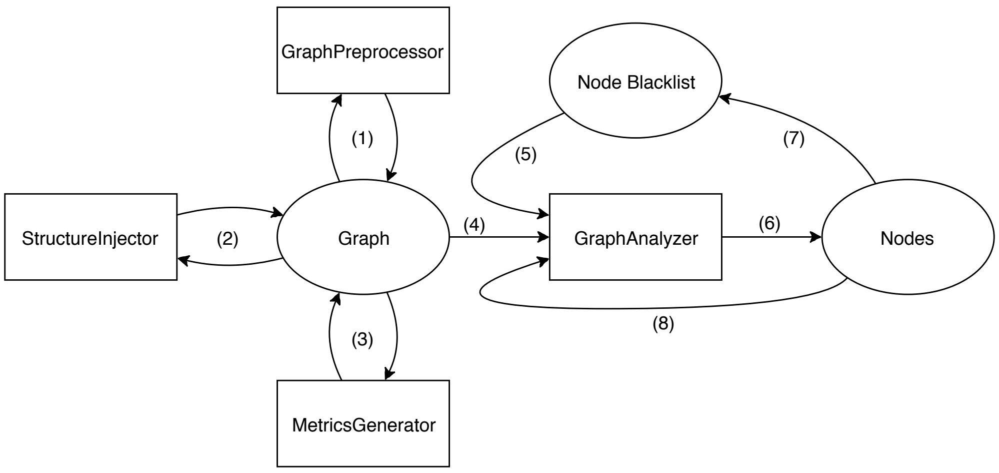

Figure 5.2: Analysis Platform

5.1.1 Graph Preprocessing

The `GraphPreprocessor` module prepares the graph contents. There are some nodes and relationships depending on the source of the graph that need to be deleted before the generation of metrics (see (1) in Figure 5.2). Initially the graph may contain nodes or relationships that are not wanted. In some cases the `GraphPreprocessor` is relevant since the subsequent metrics may be disrupted. The only relevant structures within the scope of this thesis are the mentioned nodes and relationships in 3.2.1.

Within this thesis we make use of graphs that are the result of the scanning of AD networks with SharpHound (see Section 3.2.1). Before, we explained the relationships that are relevant for this thesis in Table 3.1. SharpHound creates other relationships of other types that are not appropriate to use with regards to graph-theoretic metrics. In this work we make use of the mentioned edge types exclusively. Those additional relationship types are created by the `DBCreator` script as well (see section in 3.2.3). This is why we need to delete disturbing relationships with the `GraphPreprocessor` module.

5.1.2 Injection of Undesired Structures

The `StructureInjector` is used during the testing phase of metrics. It can be utilized to inject undesired sub-structures into the existing graph (see (2) in Figure 5.2) before the generation of metrics take place (see (3) in the diagram). Afterwards one can check whether the previously injected sub-structures can be found by using the `GraphAnalyzer` module (see Section 5.1.4).

5.1.3 Metric Score Annotation

The `MetricsGenerator` module applies graph metrics to the graph and therefore it annotates every node with the particular metric score for that specific node (see (3) in the diagram). This module is flexible regarding the usage of particular graph metric algorithms. Within the scope of this thesis we make use of the graph algorithms described in 2.5. The only requirement of subsequent analysis algorithms is that the metric algorithm is needed to annotate every individual node with the belonging metric score value.

5.1.4 Analysis of Metric Scores

The actual analysis happens within the `GraphAnalyzer` module (see (4) in Figure 5.2). Usually this module returns nodes as a result (see (6) in Figure 5.2). On the one hand, those nodes may be part of the identified sub-structures as described in the requirements (see 4.4.2). On the other hand, those nodes can be used for further processing (see (7) in Figure 5.2). Even the `GraphAnalyzer` itself is able to process resulting nodes again in order to furthermore analyze the graph (see (8) in 5.2).

This module is able to blacklist a set of nodes (see (5) in Figure 5.2). This enables the `GraphAnalyzer` to ignore those nodes in order to avoid them to appear in the returned node lists in the following iterations.

5.2 Graph Analysis Methodology

This section introduces the used methodology regarding the analysis of the network session and permission graph. Hereby, the previously explained modules are combined in a way that allows to analyze a graph regarding potentially undesired configurations.

5.2.1 Top Lists

As explained in Section 4.4.3, the graph-theoretic metrics may be used in order to discover potentially undesired configurations. All introduced centrality metrics result in higher scores when a node has a higher potential of being part of an undesired configuration. Therefore, it is an appropriate solution to fetch a list of nodes from the database. Each of those lists need to be ordered descending by the particular metric score.

Within this thesis, the most interesting nodes are nodes with unusually high scores. The investigation process in Section 4.4.1 requires further assessment of those nodes. Hence, the top scores of each metric is interesting for a further investigation. The implemented solution supports this in two strategies. The first is to query a defined amount of nodes. This approach makes sense if the underlying graph has a fixed size, meaning a particular amount of nodes and relationships. This is not the case necessarily since the size depends on the organizational structure and type of the organization.

Therefore, the designed solution supports a second more dynamic strategy additionally to the described static one. This means that the amount of nodes does not need to be specified. Instead, this value is replaced with the percentage of top scores. Hence, it is possible to fetch the top nodes that exceed a particular percentile regarding a centrality metric.

Before querying the top nodes (statically as well as dynamically), nodes may be defined as blacklisted. The next section describes how those blacklists may be applied.

5.2.2 Blacklisting

We make use of two fundamentally different blacklisting methods. The first of them utilizes statically defined blacklists. They are set up in order to block particular nodes within the graph. Those nodes need to be described explicitly tailored for the very organization. There are still a few groups that need to be ignored as stated in the requirements (see 4.4.2). However, there may be nodes that need to be ignored in multiple organizations.

The second blacklisting method is used to ignore kinds of nodes that need to be defined in more complex way. As described in the requirements (see 4.4.2) there may be nodes where it is impossible to declare them explicitly. For those nodes it is necessary to exclude them based on relationships to common nodes. Hereby we use the common semantic context to blacklist nodes in a dynamic way.

5.2.3 Graphical Analysis with Histograms

The static and dynamic blacklists explained in Section 5.2.1 provide the opportunity to identify the nodes with the highest scores with respect to particular metrics. However, it is still unclear whether those nodes are part of potentially undesired configurations. For instance, there may be a graph where all nodes are assigned with equal metric scores.

According to the reasoning given with this thesis, all nodes within this graph may be or may be no part of an undesired configuration. It would be impossible to discover extraordinary structures with the help of centrality metrics. This is the reason for an additional analysis strategy that needs to get applied within the investigation process.

Histograms are used to identify outliers regarding particular metrics. Later in the evaluation chapter, those histograms are used to get a feeling for the distribution of metric scores. Especially logarithmically scaled histograms are a comprehensive tool to find outliers with particularly high scores. Large piles of equally high scores are assumed to be part of usual structures within the assessed graph. The same applies for larger ranges of scores where nodes pop-up evenly distributed. The situation is different if a large amount of nodes reside with relatively low scores and a few nodes have high scores. This leads to the assumption that the latter ones have a higher probability of being part of an undesired configuration with respect to the defined attacker's behavior.

Chapter 6

Evaluation

In this chapter we follow two approaches in order to evaluate the solution described in Chapter 5. The first part of the evaluation is based on randomly created permission graphs to discover undesired configurations. The second part involves a real-world network that is scanned in order to get a real network session and permission graph. In both cases we compare the results with the requirements in Section 4.4.2.

6.1 Evaluation Environments

We want to introduce both environments used to evaluate the described solution.

6.1.1 Randomly Generated Graphs

During the development phase the `DBCreator` tool is used that has been developed as part of the BloodHound project (see Section 3.2.3). The following sequence shows the first approach to evaluate our solution:

1. Randomly generate graphs
2. Design undesired configuration
3. Inject undesired configuration into graph
4. Find undesired configuration again

In Step 1 a random graph needs to be generated. We make use of the `DBCreator` script for this. As described in Section 5.1.1, it is necessary to execute the `GraphPreprocessor`. Afterwards a particular undesired configuration needs to get designed in Step 2 by creating a graph sub-structure. This configuration gets injected within Step 3 into the randomly generated graph as described in Section 5.1.2. In Step 4 the described solution is used to find the previously injected configuration again.

6.1.2 Real-World Graphs

The second evaluation environment concerns real-world graphs. This evaluation is done according to the process defined in Section 4.4.1.

The impediment regarding this evaluation strategy is that it is difficult to know whether the detected configuration is actually undesired or not. We cannot find any solution to do such predictions on the findings. This is why we carried out discussions with experts within the organization in order to evaluate the meaningfulness of our results.

6.2 Evaluation of Effectiveness

As part of this thesis a framework is developed that makes it easy to arrange and inject undesired configurations. During the analysis chapter, the requirements for our solution are explained more precisely. The main features are the definition, injection and afterwards also the detection of undesired configurations.

This section describes how different configurations are designed to check the functionality of our solution. Therefore, we use the `DBCreator` of the BloodHound project to check whether our solution works on their random graphs. We point out that the generated graphs show up with similarities due to the attempt of creating realistic graphs (see Section 3.2.3). The tool generates random graphs with predefined specifications, e.g. a graph always contains an IT department and administrators. Such schemes need to be followed in order to simulate an AD graph of a real organization. This is the reason, why the following sections contain average results. The generated graphs show up with hardly distinguishable results regarding injected sub-structures. Hence, the results are deduced from three experiments each. The results are calculated as the average value of those three experiments each.

The used workflow for the simulated environment is the following: At first we generate a new random network session and permission graph in the database. Afterwards the described modules from the Design and Implementation chapter (see Section 5.1) are arranged appropriately in order to inject and find particular graph sub-structures again.

The modules are arranged as follows:

1. `GraphPreprocessor` deletes nodes and edges that would disturb the generation of graph-theoretic metrics
2. `StructureInjector` injects a predefined undesired configuration that needs to be found afterwards
3. `MetricsGenerator` generates predefined metrics, e.g. Betweenness Centrality or Closeness Centrality
4. `GraphAnalyzer` fetches nodes that need to get excluded from further analysis steps
5. `GraphAnalyzer` fetches the calculated metrics and uses them to collect statistics or even visualize the results

Afterwards we take the output data from the graph analyzer in order to examine them.

For the real-world scenarios, the first two modules in the list are not necessary. This is because the graph of the present case does not include sub-structures that need to be excluded from the analysis. Obviously, the `StructureInjector` is not needed for the analysis because potentially undesired configurations need to be discovered in the real-world scenario.

The blacklists are created with the following strategy. The analysis platform is equipped with similar blacklists for the real-world and simulated scenarios. There is a list of default security groups that is utilized, created by Microsoft [11]. Those security groups are needed within the AD for technical reasons and they are created automatically when an AD network is created. For instance, this includes the `DOMAIN ADMINS`, the `DOMAIN USERS` and the `DOMAIN COMPUTERS` groups. Accordingly, they consist of domain admin users, all domain users and all domain computers as members. These groups are excluded from the analysis sequence completely by using the priorly defined static blacklisting mechanism. The dynamic blacklisting mechanism described in the Design and Implementation chapter 5 is used to exclude all users that are part of the domain admins group as well. The reason for this is that we expect high centrality scores for domain admin users since they have typically they have an unusually large amount of admin permissions throughout the AD network.

6.2.1 Common High-Value Targets

We fetched lists of the nodes ordered by the particular centrality metric scores in a descending manner. Common high-value targets within AD like the domain admins group are listed on the very top of those lists for real-world an simulated scenarios.

This evidence shows that the centrality metrics have the potential to find high-value targets that are not as common as the default AD groups as well.

Groups like the domain admins group are necessary in order to keep users that are able to maintain the AD network. On the one hand, this group may get blacklisted within our solution. The reason is that the domain admins group is rather common and it cannot be avoided because of technical causes. On the other hand it may be interesting to assess such security groups on a regular basis, since such groups may be beneficial for an attacker in particular.

6.2.2 Computers with Multiple Administrators

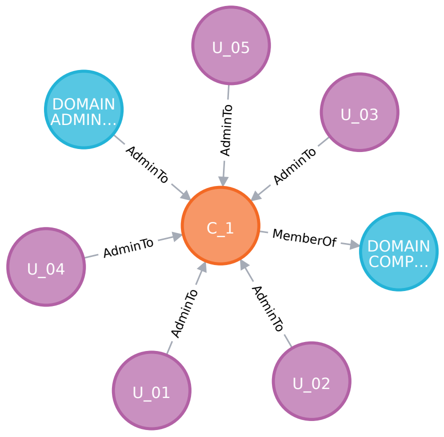

Figure 6.1: Undesired configuration: One computer with five admin users

This section is about the injection of computers where an unusual amount of users have administrative privileges on. This means that one computer needs to get injected to simulate this. Furthermore, an untypical amount of users need to get administrative privileges onto that computer by using the `admin-to` relationship.

A scenario like this would be beneficial for an attacker because of two reasons. On the one hand, an attacker would get higher profit from compromising the computer. If the computer is compromised, the attacker has many options regarding stealing identities because there are more admin users for this computer than usual. On the other hand, an attacker has an unusually high amount of paths to the computer. It would be enough to steal one identity of the admin users to enable lateral movement to the computer for further compromising.

Figure 6.1 shows the described configuration. The mentioned computer can be seen in the middle (orange circle). The purple circles around it depict five test users that have administrative privileges on the computer. The blue circles depict two general groups (as explained in Section 6.2.1). The computer is a member of the `DOMAIN COMPUTERS` group as usual in AD. The `DOMAIN ADMINS` group has administrative privileges on the test computer. This relationship is typical for most AD systems as well.

In total we use a generated graph including 508 AD groups, 505 users and 511 computers. Those nodes are connected with 2877 `member-of`, 506 `has-session` and 2289 `admin-to` relationships.

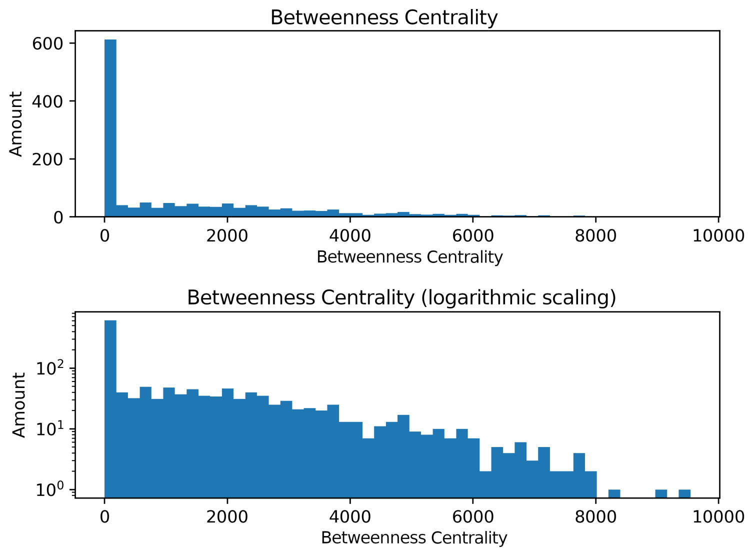

Figure 6.2: Betweenness Centrality of a graph (injected one computer with five admin users)

Figure 6.2 shows the distribution of the Betweenness Centrality score. This and all following histograms make use of 50 bins each. The underlying reasoning is that the important parts of the histograms are the nodes with the highest scores that are far off from accumulations at lower scores. Hence, this decision does not impair the particular analysis results, even if there are striking accumulations on the left-hand side of a histogram. This means that the crucial results are the outliers on the very right side of the histogram. All histograms show the distribution with normal scaling on the top. On the bottom a logarithmic scaling is applied to increase the recognition of the score distribution.

In Figure 6.2 it can be seen that there is a clear accumulation of relatively low scores between 0 and under 8000. However, the previously injected computer has a score of 3083 and therefore it is hard to use the Betweenness Centrality as an indicator in this case.

A possible explanation for this behavior of the Betweenness Centrality score is that there may be multiple similar sub-structures. This would suggest that the node cannot be discovered within the given graph because of other usual sub-structures that show up with similar connections within the graph.

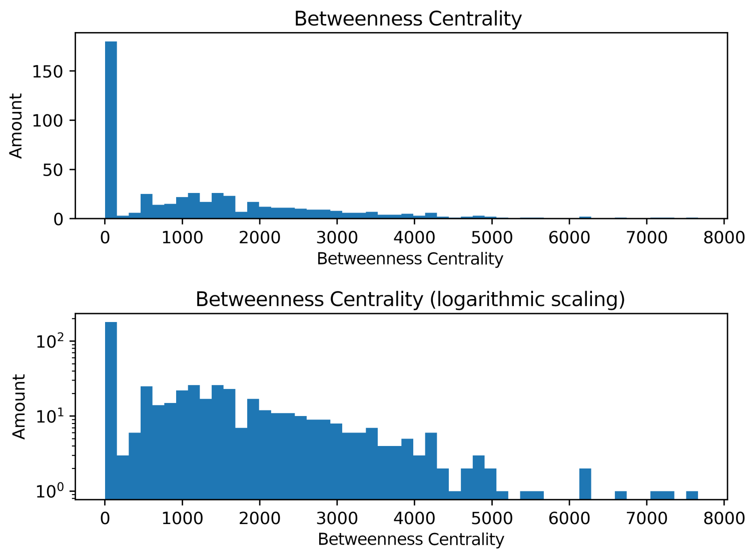

Figure 6.3: Betweenness Centrality of the graph (injected one computer with ten admin users)

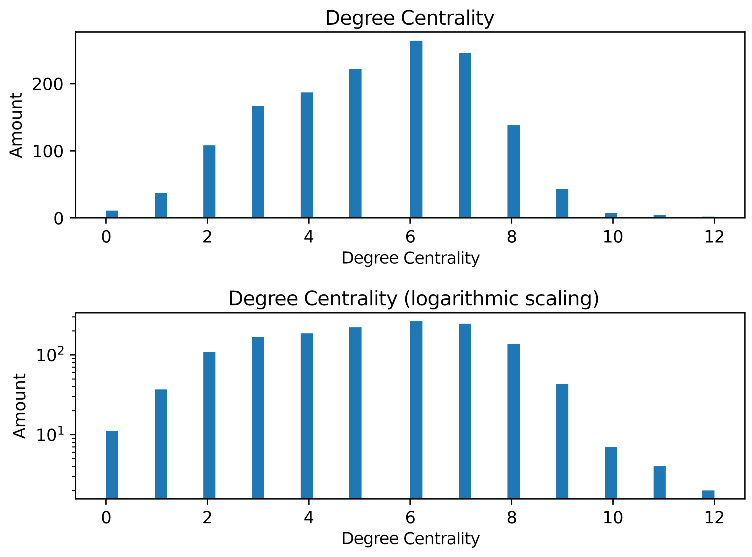

Figure 6.4: Degree Centrality of a graph (injected one computer with ten admin users)

The Degree Centrality score supports this reasoning because the test computer has a score of seven for this metric. This is a rather usual score within the graph as well.

Figure 6.3 shows the distribution of the scores if the admin users are increased to the amount of ten identities. For this case the usage of Betweenness Centrality seems more useful since the score of the injected computer is now at about 7660. However, only twelve nodes have a higher Betweenness Centrality score than this For a security team it would be a tolerable affort to check this little amount of nodes until the discovery of the test computer. Hence, despite the size of this graph, it is still possible to detect the injected test computer efficiently with this amount of admin users on a single computer.

Now, we take a look at the Degree Centrality score again to check the scenario regarding unusual structures. Figure 6.4 shows the Degree Centrality score distribution for one test computer with ten users with administrative privileges. There is a striking accumulation of nodes that obtain a score of up to nine. Higher scores are very seldom for this graph compared to other bins in the histogram. As for the Betweenness Centrality it is possible to discover the injected test computer since it has twelve neighbors in total (ten users and two groups as explained before). In this scenario, this means that the test computer has the highest Degree Centrality score.

In summary, the Betweenness Centrality and the Degree Centrality fit well for the given scenario. It is possible to find undesired configurations as previously defined in this section. One last experiment shows how the usage of both metrics at once improves the results again. We queried the highest 1% of all scores of both metrics. Then an intersection of both sets leads to only one last node as a result, which is the previously defined test computer again.

This leads us to the conclusion that the Betweenness and Degree Centrality are both well fitting metrics to uncover the described undesired configuration. Especially the combination of both metrics leads to even better results.

Section 4.4.3 describes several behaviors of an attacker and how they may match with centrality metrics. Hereby we describe a scenario where two of the attacker's behaviors apply in respect of Betweenness and Degree Centrality scores. On the one hand, the attacker may profit from the direct neighborhood of particular nodes during lateral movement. On the other hand, a large amount of shortest paths the attacker may use cross the mentioned test computer. Hence, the injected configuration is beneficial for the attacker's lateral movement.

During those experiments, we considered Closeness Centrality as a well-performing metric as well. It turns out, that none of the nodes in Figure 6.1 nor additionally associated test user nodes can be discovered by searching for high centrality scores.

6.2.3 Users with Extensive Administrative Privileges

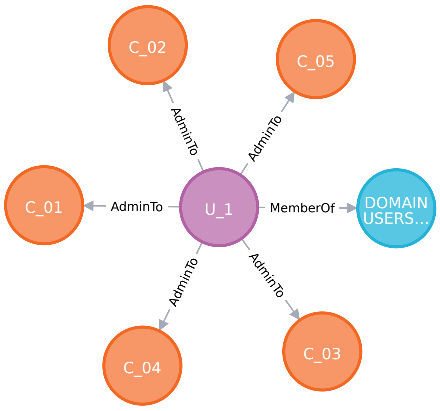

Figure 6.5: Undesired configuration: One user with admin privileges

We assume the next configuration to be undesired as it depicts an highly privileged user. Figure 6.5 shows one user in the middle that has administrative privileges on five computers in total (orange circles). Additionally the user is part of an usual group that contains all AD users (`DOMAIN USERS`).

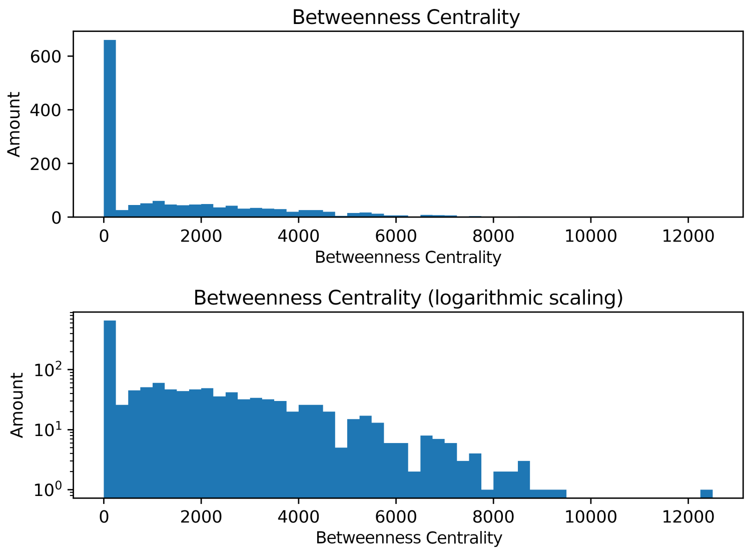

Figure 6.6: Betweenness Centrality of a graph (one user as admin of five computers)

This configuration may be undesired because of two reasons. First, compromising the user U_1 is beneficial for the attacker's further lateral movement since there are many possible paths to the surrounding computers. This is depicted with a single step via the admin-to relationship to the particular machines. Second, the probability of passing U_1 during a random walk of the attacker is rather high because there are many computers that may be used for this. If one of the surrounding computers is compromised by the

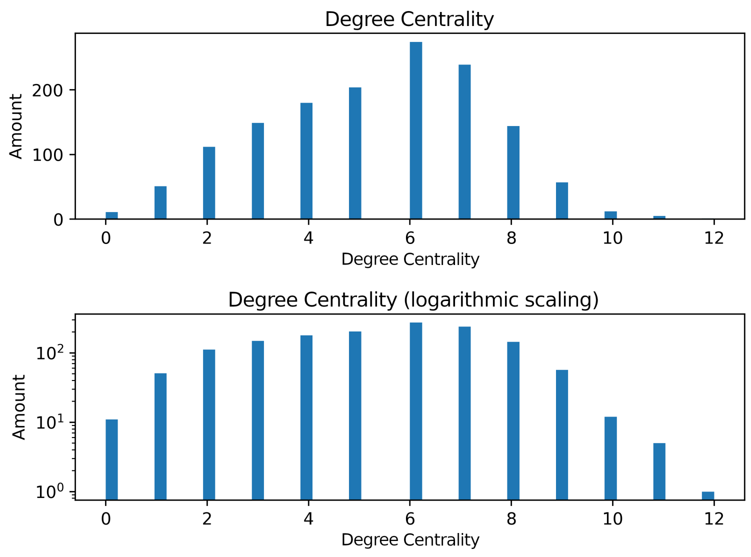

Figure 6.7: Degree Centrality of a graph (One user as admin of ten computers)

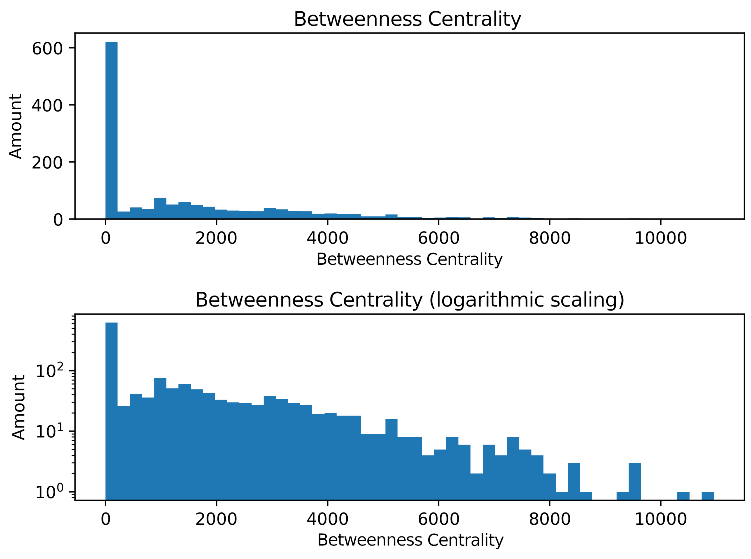

Figure 6.8: Betweenness Centrality of a graph (one user as admin of ten computers)

attacker, he may make use of the direct neighborhood. Therefore this configuration `U_1` is declared to be undesired.

Here we generate a similar graph as before with the `DBCreator`. It consists of 508 groups, 515 computers and 501 users in total but the user in Figure 6.5 obtains a Betweenness Centrality score of about 5581. The histogram in Figure 6.6 shows that this value is one of the highest scores in the entire graph. In fact, the ordered top-1% list of all Betweenness Centrality scores shows that 49 other nodes are positioned before the injected user. For a graph of that size it still might be appropriate to check the first 50 nodes in that list but it looks like the injected structure is not that unusual in the entire graph. The Degree and Closeness Centrality top lists neither provide the particular node with a top score.

Increasing the computer count shows more positive results since this seems to be a more unusual structure. The user now pops up even in the top-5% node list of the Degree (Figure 6.7) as well as of the Betweenness Centrality scores (Figure 6.8). In both lists it has a high score with a Degree Centrality of eleven and a Betweenness Centrality of 10955. Further experiments show that the structure still shows up on the eleventh place of the Betweenness top list if the admin privileges are reduced to 7.

6.2.4 Domain Administrator's Subgroup

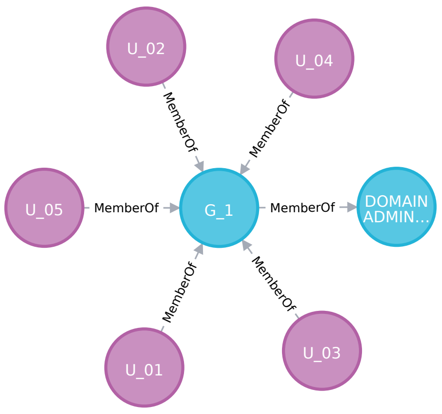

Figure 6.9: Undesired configuration: Domain administrator's subgroup with five users

The next undesired configuration injected into random graphs can be seen in Figure 6.9. This configuration consists of a group, that belongs to the `DOMAIN ADMINS` group. The problem here is that the `DOMAIN ADMINS` group is a functional AD group that has administrative permissions through the entire domain. Hence, the injected group `G_1` in the configuration profits from the same permissions as the functional group. That means that each and every member of group `G_1` may administer every computer within a typical AD network. In respect to this undesired configuration, all purple users in Figure 6.9 are affected.

This configuration is undesired because an attacker may profit regarding his behavior as described in Section 4.4.3. Group `G_1` needs to get passed on every shortest path between one of the members of the group and some computer in the network. Therefore this group is a higher risk regarding lateral movement of the attacker. Furthermore it may be enough if one user of the group gets compromised to reach full permission on the whole network. This is a crucial threat since after that step an attacker may compromise a high amount of other computers within the AD. One last risk is that there are many memberships regarding group `G_1`. This means that the group may help the attacker during an Identity Snowball Attack to gain higher privileges. Therefore, the probability of many compromises is higher.

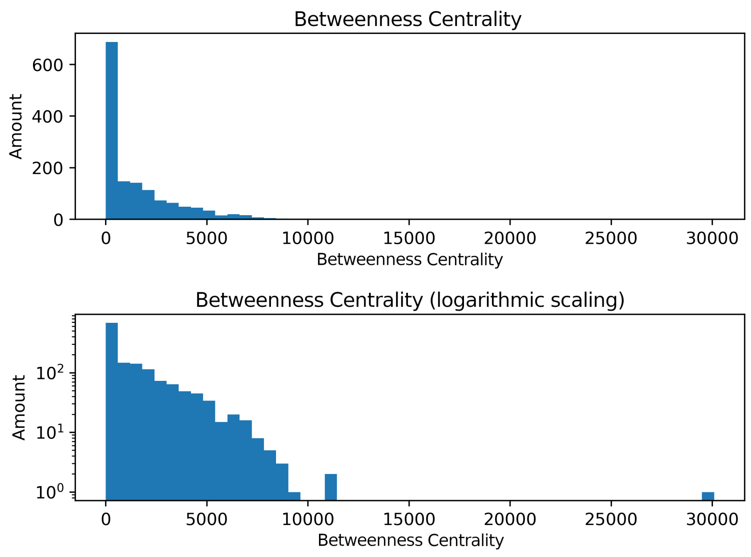

Figure 6.10: Betweenness Centrality of a graph (domain admins' subgroup with five users)

Figure 6.10 depicts a histogram that shows the distribution of the Betweenness Centrality scores throughout the AD network. Especially the logarithmic scaling (histogram below) clearly shows that there is an easy to recognize high amount of scores below 8000. The score of `G_1` is about 11489. This is the node with one of the highest scores for that centrality metric and therefore a node that is easy to discover in an automatic manner.

As can be seen in Figure 6.9 the Degree Centrality score of the group is six. This score is too low to stand out within the whole graph.

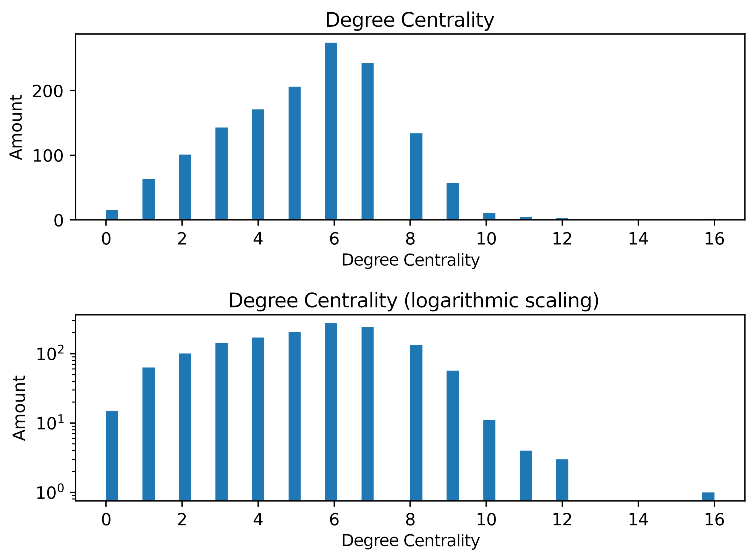

Figure 6.11: Degree Centrality of a graph (domain admins' subgroup with ten users)

Figure 6.11 shows the Degree Centrality score distribution when the undesired configuration gets injected with ten users as members of the group. The score of `G_1` is eleven now. Only one node has a higher score than eleven and therefore `G_1` may be detected with low effort for this case.

As a result, it can be seen that this configuration cannot be discovered with an amount of five users. It seems like this is still an usual configuration within the graph. Hence, e.g. ten admin users are detected as an unusual and therefore an undesired configuration.

6.2.5 Analysis of an Existing Active Directory

As mentioned already, this thesis evaluates whether the introduced metrics would support the investigation process described in Section 4.4.1. Therefore the scanning is done in order to collect the graph data (Step 1 of the process). This means that the existing AD network is scanned with 3.2.1 in order to persist the users, groups and computers with the described relationships. Afterwards the previously introduced metrics are applied in order to find configurations that may be beneficial regarding the attacker's behavior. Section 4.4.3 explains the behaviors we want to tackle with this strategy utilizing the centrality metrics.

This section may give some insights into the analysis utilizing the investigation process. We demonstrate how it may be possible to discover undesired configurations in real AD networks that are significantly larger than the generated graphs.

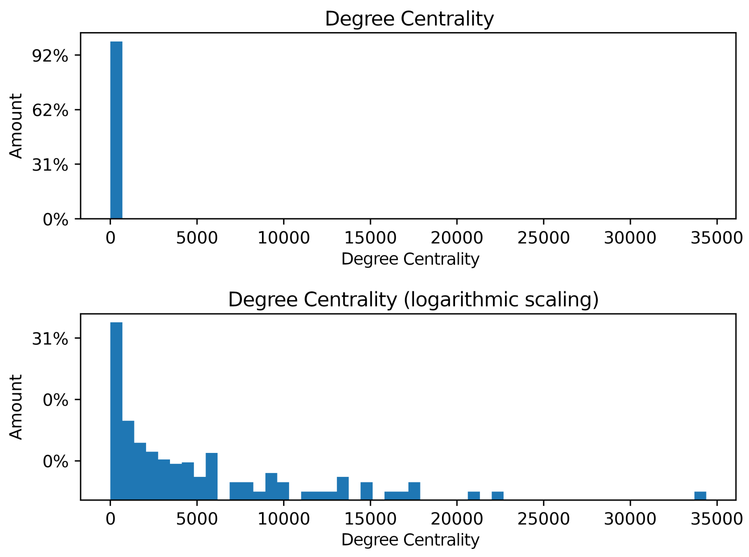

Figure 6.12: Degree Centrality distribution in a real AD network

The degree centrality distribution can be seen in Figure 6.12. The distribution regarding random graphs clearly diverges. The reason may be that the graph bases on other organization structures than the randomly generated graphs are based on. Referring to one of the described attacker's behaviors, the probability of compromising increases for higher degrees because of more entry points to a node. In this figure, it can be

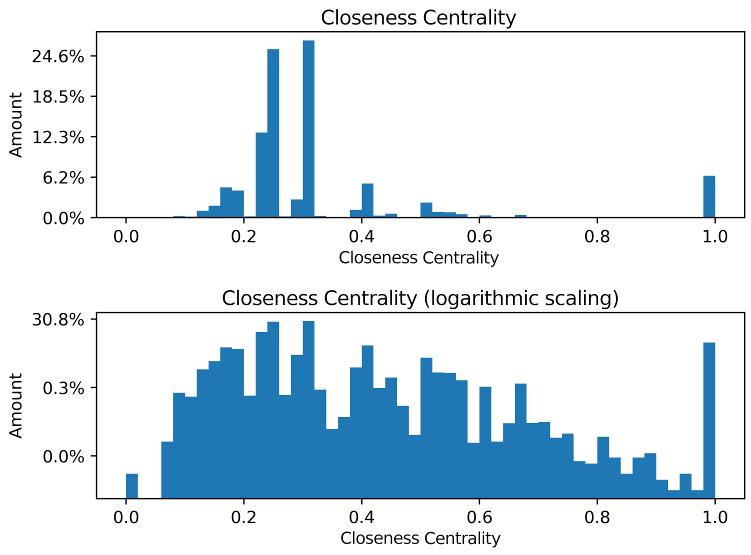

Figure 6.13: Closeness Centrality distribution in a real AD network

seen that a large number of nodes have a low number of neighbors. Despite of this, the histogram with the logarithmic scaling at the bottom, clearly shows that there are nodes with more than 30000 neighbors. This seems to be very untypical for the AD graph and therefore those nodes are required to get examined more precisely.

The next histograms in Figure 6.13 shows the distribution of the Closeness Centrality scores. The last sections in the evaluation chapter suggest that the Closeness Centrality is not as appropriate to find undesired configurations as other metrics. Nevertheless, anomalies may be detected within the real AD graph. A Closeness Centrality score of approximately 1.0 means, that such a node is connected directly to almost all nodes within the graph. We want to emphasize this result since it affects more than 6% of the graph, despite this is no anomaly relating to the described attacker's behaviors. Hence, an attacker may be closer to all nodes on average and therefore he may profit more from those nodes than from others.

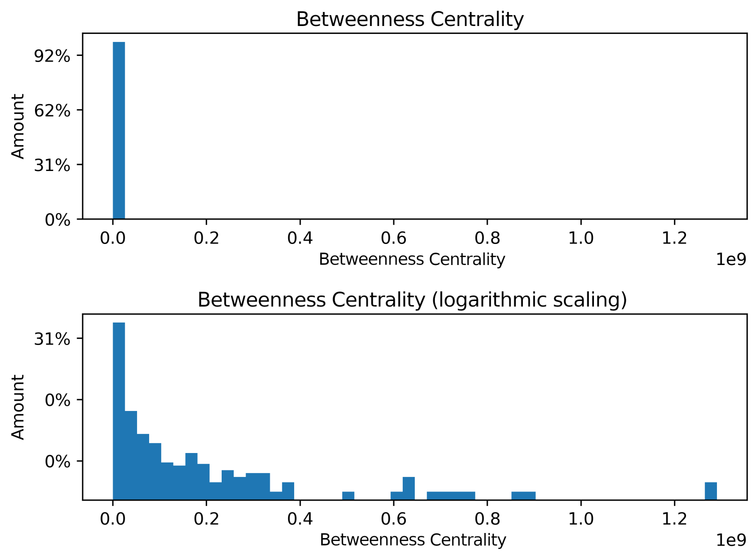

FIGURE 6.14: Betweenness Centrality distribution in a real AD network

The last score distribution is about the Betweenness Centrality. Figure 6.14 depicts the belonging histograms. As the metrics before show as well, this centrality metric shows up with an accumulation at lower scores. Still, there are some extraordinary scores as can be seen on the right side of the lower histogram. Derived from the graph-theoretical definition, this means that there are still nodes that lie on many shortest paths within the graph. With regard to an attacker, this means that those nodes may have a higher probability of getting compromised than others. Hence, this set of hosts would be worth for close investigation.

In summary, we can state that all introduced metrics deviate from normal distributions. They provide insights into nodes that are outliers regarding different centrality metrics. The hereby identified amount of nodes is too high to do further investigation in detail. Therefore, within this thesis it is possible to take only a first look at Step 3 after the graph-theoretical analysis. This step may contain discussions with several users, system administrators and more stakeholders in order to find out how high centrality scores may be avoided in the future. Nevertheless, we are able to achieve a few insights.

At first, common high-value targets need to get mentioned as described in Section 6.2.1. The security groups created by default are not blacklisted at the beginning of the analysis. The reason is that discovering them is an indication for the operation of the metrics. As presumed, the `DOMAIN ADMINS`, `DOMAIN COMPUTERS`, `DOMAIN USERS` and other groups defined by Microsoft can be found. Thus, this result is the same as for the generated random graphs. After excluding such groups by appending them to the blacklists, the analysis continues with rather non-standard nodes.

One exemplary finding depicts a user that is a member of a large amount of different groups within the AD. Those groups are not part of the default groups mentioned in the last paragraph. A configuration like this might have different reasons that have not been worked out until now particularly. One example may be that the user switched positions within the organization many times or worked with other people in different environments. This may have lead to the membership in an unusually large number of groups. This might be an undesired configuration because a potential attacker may profit from the user with its memberships. On the one hand, this leads to a high number of neighbors of the user node. On the other hand, this user may be on a large number of shortest paths. This would make sense since both Degree Centrality and Betweenness Centrality are higher than usual for this node. The reasons for this configuration and possible mitigation strategies, further manual investigation needs to be done in the future.

Other exemplary findings turn out to be functional user accounts. All top lists – the scores of Degree, Betweenness and Closeness Centrality – contain a large number of users that are used for technical issues within the network. Such users typically obtain unusual amounts of memberships to other nodes. On the one hand they may automatically establish sessions. On the other hand they may require many permissions, assigned via direct administrative permissions or with transitive permissions via group memberships. This is the reason for large numbers of direct and transitive relationships and therefore high scores regarding the mentioned metrics. We suggest to further investigate such functional accounts in order to sort out those that are no longer required.

Furthermore there may be functional accounts, where a decreased amount of privileges may be enough. Such further steps may be done in the future as well.

During the analysis, new analysis results are assessed as explained in the investigation process. When a node shows up with a high metric score and it does not need to be proposed within further iterations, it simply gets excluded by adding it to the blacklist. This is how the investigation process avoids unnecessary effort regarding multiple assessments of the same sub-structure.

The previous sections state how undesired configurations can be detected in randomly generated graphs. For better comparison, those undesired configurations are injected into the real AD graph. Hereby, all injected configurations can be discovered in a similar manner. A computer is found if an unusual amount of users have administrative permissions for it. As for the random graphs, this configuration is found if there are ten admin users. The same applies for a user that has administrative permissions to ten computers. A sub-group of the `DOMAIN ADMINS` group is discovered if it contains five users. All of those configurations can be detected explicitly when the top 1% percentile lists of Betweenness Centrality and Degree Centrality get intersected.

6.3 PageRank as a Comprehensive Metric

In the chapters before, the PageRank metric is not used. The reason for this is that we are not able to define a specific attacker's behavior suitable for this metric. However, there are some clues that indicate that PageRank is a comprehensive metric to discover undesired configurations.

The PageRank is applied onto all mentioned scenarios described in this chapter. In some experiments PageRank increases accordingly to other metrics we introduce before. Nodes can be discovered as undesired configurations because of high metric scores. Most of those discoveries incidentally correspond to high PageRank scores as well. Hence, it seems that PageRank is a well-suited indicator for undesired metrics.

In some experiments it is enough if one metric increases. The PageRank instantaneously increased accordingly. Regarding injected configurations into random graphs, there are strong correlations between the PageRank and the Degree, Closeness and Betweenness Centrality. Despite of that, the last three metrics hardly ever correlate with each other. This leads to the assumption that the PageRank metric is a combination of metrics that unites the top lists comprehensively. It may be possible to use the PageRank instead of all other metrics in order to have a one generic metric. This method may lead to a simplification of the overall process since only one metric would be necessary.

One example is described in Section 6.2.2, where a computer is injected that has an untypically large amount of admin users. Here the PageRank score of the affected computer is high along with the Betweenness Centrality score. Furthermore the PageRank even exposes a computer with only five admin users, which is not possible with any other centrality metric. Hence, the PageRank metric may leverage the investigation process to even expose inconspicuous undesired configurations.

There are correlations between the PageRank and other metrics regarding the real AD network as well. Thereby, the correlations strongly affect the Degree Centrality and Betweenness Centrality. The Closeness Centrality correlates with the PageRank as well but less pronounced. However, the intersection of the 1% percentile top list of the PageRank and the Closeness Centrality still consist of one node that reaches a high score with both metrics.

6.4 Conceptualization of Undesired Configurations

The past sections clearly depict that configurations may be undesired out of a subjective view only. It depends on the particular graph and therefore it depends on the organizational and technical structure. Specifically defined and injected configurations show up with one parameter at least in all introduced scenarios. If we increase such a parameter, e.g. the amount of admin users in Section 6.2.4, the configuration becomes undesired. How much this parameter needs to get increased depends on the AD generated structure.

In the described scenarios, we describe how to find the injected structures. There is no guarantee that there are no other sub-structures within the graph that may be undesired as well. This leads to the assumption that every usage of the introduced methodology needs to be carefully studied in order to come up with meaningful thresholds of every score top list.

A pretty similar reasoning may be applied to the real-world evaluation in Section 6.2.5. The investigation process always needs to get applied with the organizational scope in mind. Therefore, the hereby described findings depend on the structure of the organization.

6.5 Comparison to Related Work

Chapter 3 refers to the Heat-ray project and the BloodHound tools as related work. Within this section we compare them with the introduced solution of this thesis.

6.5.1 Heat-ray

The primary goal of Heat-ray is the detection of bridges between large components of the network session and permission graph. The authors want to cut those bridges in order to mitigate the risk of an attacker passing them. The reasoning behind this is that cutting them would increase the security of the whole AD network. If an attacker successfully compromised one large component, it would not be possible to do lateral movement to the other component. However, the second large component remains secure.

In summary, this means that Heat-ray considers one undesired configuration – bridges between large components. The solution developed within this thesis considers them undesired as well. We utilize Betweenness Centrality in a similar manner than Heat-ray in order to describe the attacker's behavior as Heat-ray does. However, the solution within this thesis provides additional metrics as explained in 4.4.3. Therefore our solution may consider multiple possible behaviors of an attacker by using those metrics.

It may be possible to integrate the centrality metrics within this thesis into the Heat-ray approach in order to refine the results. This approach may enable the Heat-ray process to cover diverse attackers' behavior. This would result in minor changes within the Sparsest Cut algorithm resp. its objective function of the linear programming.

Heat-ray makes use of two methods to find important sub-structures that should be considered to get proposed to an IT administrator. The first one is to calculate the amount of shortest paths that pass an edge. The second one includes an approximation algorithm in order to save performance. Within our solution those approximation are not necessary because all metrics can be used in an appropriate period. It is still possible to calculate the metrics through the entire graph even for organizations with large networks. We make tests with networks including large amounts of nodes and relationships. The developed solution is still usable interactively since the metrics still finished within minutes on a desktop machine.

Another difference of our solution compared to Heat-ray is how the approaches deal with unwanted proposals of the algorithm. Initially, Heat-ray does not make use of any blacklisting methodology. After the first iteration, it suddenly uses the feedback of some IT administrator in order to decrease the probability of proposing an edge again. However, this does not hinder the algorithm to propose the same edge again but the probability that this happens is rather low. Heat-ray applies a Machine Learning algorithm in order to learn from the administrator's feedback. This is how they use the feedback not only for the affected edges, but for additional edges within the graph.

For simplicity, we make use of easier concepts by utilizing blacklists as explained in Section 4.4.2. This solution works well in a similar manner than the Heat-ray approach. The feedback can take affect on the future blacklists for further iterations. This is how our solution avoids additional proposals of sub-structures by completely excluding them from the analysis iterations.

A detailed comparison between Heat-ray and the hereby introduced solution is not possible because the Heat-ray implementation is not available.

6.5.2 BloodHound

The second introduced related work in Chapter 3 is the BloodHound project. Thereby we presented the BloodHound features within the provided tools. An analysis with the BloodHound user interface contains manual workload for most use cases. BloodHound is developed for manual analysis despite it is a centralized approach. This means that an analysis with BloodHound happens at a central point, i.e. the analyzing staff may use the AD filled graph database from a central point. The drawback here is that analysis methods are limited. The analyzing staff needs to specify the target system they are interested in. This ends in an high effort if one may want to analyze any number of machines within the network.

The effort gets even higher if another parameter of the analysis are systems that may be compromised additionally to the target system. The defensive perspective may require an analysis of all possible systems in the role of a target system and furthermore in the role of a compromised machine. This ends in staff analyzing every possible path within the graph, i.e. the algorithmic complexity increases the manual effort for the user again.

An automatic analysis with BloodHound is only possible if the analysis conditions are set very precisely. Therefore, automated analysis is restricted to a limited number of use cases. One example may be the finding users or groups with an extraordinary amount of `admin-to` relationships. Hence, this feature would reveal users with administrative privileges for an high amount of computers.

Overall, we found that BloodHound is not able to do automated analysis that results in proposals of particular undesired sub-structures within the graph that are independent of the AD context. In contrast to this, the introduced solution regarding this thesis supports security staff by automatically exposing undesired nodes within the graph. Knowledge about the context is not necessary because the presented metrics are designed from a more general perspective with graph-theoretic approaches. This allows us to abstract our approach to general network session and permission graphs. Thus, our solution covers not only AD networks as the BloodHound project does.

Chapter 7

Future Work

This thesis provides methods to analyze network session and permission graphs that can be expanded to cover further objectives. Those goals may lead to further improvements of protection against lateral movement. This chapter presents ideas of how to expand or reuse the developed approaches.

There are several possible enhancements for the designed approach. Reference should therefore be made to Section 6.3 as correlations show that PageRank may be a well-suited metric to identify undesired configurations. Additional steps are necessary to ensure that PageRank is a kind of all-inclusive metric. There is some evidence that PageRank depicts the interaction between particular characteristics that can be captured by the introduced metrics. Therefore, it is necessary to define attacker behaviors that fit the graph-theoretic method of operation of the PageRank. This may explain why the PageRank provides such accurate results according to other metrics.

A continuation of this thesis may also be to reuse particular evaluation results. The introduced investigation process (see Section 4.4.1) can be used to find undesired configurations in an organization. During the application of this process, the organization's security team may notice that similar undesired configurations pop up during the iterations. Hence, they could use these similarities to build up action plans as guidelines for future changes to the network session and permission graph. This approach may even result in an abstract framework that supports the further maintenance of such graphs. This can be made up of guidance for administrators for instance.

It may also be possible to design a framework for security teams that gives them the opportunity to analyze changes within the graph. Such a framework may enable the team to do temporal analysis of changing configurations in case of an incident. Those changes may be the effects of the considered attack and therefore alterations have to be investigated afterwards. Sub-structures within the graph would alter if an attacker was able to create or delete relationships within the graph. New edges within the graph would affect the calculation of metrics as well. Hence, the attacker's activities may get detected and localized by comparing metric scores of the graph before the attack with the graph afterwards.

Different edge weights influence the centrality metrics. Using additional information in order to set edge weights may therefore improve the significance of metric scores. Nodes with higher scores may be have a higher probability of being part of potential attack paths. There are several possible sources for such edge weights, e.g. the closeness to network gateways may be used. Different weights may also be used depending on the security level of a system. Therefore, systems that store crucial information may be rated differently from systems that are used for more uncritical purposes. This approach would take additional information on the value or purpose of the affected node into account.

Another continuation may be to use the approaches of this thesis together with optimization algorithms as the Heat-ray project did. Heat-ray uses only one centrality metric as described in Section 3.1. The introduced metrics of this thesis may be used as part of the Heat-ray algorithm in order to improve their strategy. PageRank may be an appropriate metric since it seems to be a comprehensive metric to discover undesired configurations.

Chapter 8

Conclusion

This thesis presents a solution that supports security teams in decreasing the risks of attackers' lateral movement within AD networks. Generally speaking, the solution is built to discover potentially undesired configurations in network session and permission graphs.

Previous solutions that tackle similar problems have a number of drawbacks. On the one hand software solutions are not able to prevent credential theft as deploying such tools is not generally possible. Especially problems occur regarding large organizations since those solutions have specific requirements that cannot be met necessarily. Maintaining permission structures in a security-aware manner is hardly possible for large organizations as there are large numbers of stakeholders. Taking all interests into account is a complex task and therefore this solution is particularly difficult to implement. Even the mitigation through network traffic monitoring proves to be unsuccessful as malign and benign network traffic is hardly distinguishable. The reason for this is that only normal API calls are used to execute Identity Snowball Attacks. The same type of traffic is produced by the behavior of normal users. Hence, those solutions are not applicable in general.

We hypothesize that an analysis of a graph-based representation of network session and permission graphs allows security teams to discover sub-structures that may be beneficial for particular attacker behaviors. This thesis designs graph configurations that may be beneficial for typical attacker behaviors regarding lateral movement. Therefore, those graph configurations may be interpreted as potentially undesired as the attacker may profit from them. We implement a graph-theoretic solution that utilizes common centrality metrics to detect the mentioned potentially undesired configurations.

To achieve this, we defined a novel investigation process that gets applied iteratively. The network session and permission graph gets collected with the help of existing tools. Then different centrality metrics are calculated for each node in the entire graph. The resulting metric scores are analyzed with descending top lists and histograms. Substructures may get excluded with blacklists to avoid results that seem to be obvious or that are assessed already.

The developed solution is evaluated in two different ways. On the one hand, the solution is tested with randomly generated graphs, inspired by typical organization structures. Several undesired configurations are injected into those graphs. Afterwards the hereby introduced solution is used to discover the injected configurations. It shows that the implemented solution is able to discover potentially undesired configurations. On the other hand, an existing AD graph is used to confirm the correct functioning of the introduced solution. This assessment shows how the approach performs in a real-world environments as well, even if this graph is significantly larger.

It turns out that Betweenness Centrality and Degree Centrality seem to be well-suited metrics to discover the defined undesired configurations. The Closeness Centrality does not perform as well as the others. It can be observed that another metric – the PageRank – presents correlations regarding other metrics. Hence, the PageRank may be examined as a promising metric in the future.

In summary, there are several advantages of the implemented solution. First, the effort of analyzing a network session and permission graph can be kept low due to its central nature. After the underlying information is collected and persisted as a graph, the analysis can take place from a central location. Furthermore, the evaluation of the real-world environment shows that the approach is scalable as the real graph is significantly larger than the generated graphs. Despite of the very specific data sources of the graph, no specific knowledge about the graph is necessary as the approach is based on general graph theory. No technical or organizational context is necessary to do the analysis. This is a major difference to the BloodHound project for instance. Hence, the approach is generally applicable to other network session and permission graphs as well. Additionally, in theory the introduced solution operates with any number of metrics at the same time other than the approach by Heat-ray. Three metrics are tested, whereby two of them are able to discover potentially undesired configurations. The Heat-ray project is limited to one metric.

This thesis shows how to detect potentially undesired configurations with graph-theoretic metrics. However, there are several opportunities to strengthen the approach with further centrality metrics. Additionally, there may be enhancements that comply with other purposes to improve network security, especially with respect to the mitigation of Identity Snowball Attacks.

Chapter A

List of Acronyms

AD	Microsoft Active Directory V, 2, 7, 9, 13, 15–17, 19, 20, 22–25, 32, 38–41, 44, 46–55, 59, 60
ATT&CK	Framework to describe Adversarial Tactics, Techniques and Common Knowledge after compromise 6
DC	Active Directory Domain Controller 7–9, 16, 20
GPO	Group Policy Object 7, 8
IDS	Intrusion Detection System 2, 21
IPS	Intrusion Prevention System 2, 21
LDAP	Lightweight Directory Access Protocol 7, 9
OIDC	Open ID Connect 6
RBAC	Role-Based Access Control 20

Bibliography

[1] John Lambert. *Defenders Think in Lists. Attackers Think in Graphs. As Long as This Is True, Attackers Win. – Defender Mindset.* Apr. 26, 2015. URL: https://blogs.technet.microsoft.com/johnla/2015/04/26/defenders-think-in-lists-attackers-think-in-graphs-as-long-as-this-is-true-attackers-win/ (visited on 05/08/2019).

[2] Hasherezade. *BadRabbit: A Closer Look at the New Version of Petya/NotPetya.* URL: https://blog.malwarebytes.com/threat-analysis/2017/10/badrabbit-closer-look-new-version-petyanotpetya/ (visited on 05/08/2019).

[3] John Miller et al. *Petya Destructive Malware Variant Spreading via Stolen Credentials and EternalBlue Exploit « Petya Destructive Malware Variant Spreading via Stolen Credentials and EternalBlue Exploit.* June 27, 2017. URL: https://www.fireeye.com/blog/threat-research/2017/06/petya-ransomware-spreading-via-eternalblue-exploit.html (visited on 05/08/2019).

[4] *Credential Stealers: Mimikatz.* Oct. 11, 2018. URL: https://cyber.gc.ca/en/guidance/credential-stealers-mimikatz (visited on 05/12/2019).

[5] Benjamin Delpy. *Mimikatz: A Little Tool to Play with Windows Security.* URL: https://github.com/gentilkiwi/mimikatz (visited on 04/25/2018).

[6] *Final: OpenID Connect Core 1.0 Incorporating Errata Set 1.* Nov. 8, 2014. URL: https://openid.net/specs/openid-connect-core-1_0.html (visited on 05/07/2019).

[7] Claudia Eckert. *IT-Sicherheit: Konzepte - Verfahren - Protokolle.* Berlin: De Gruyter Oldenbourg, 2018. ISBN: 978-3-11-055158-7.

[8] *The MITRE ATT&CK Framework: Lateral Movement.* The State of Security. Sept. 10, 2018. URL: https://www.tripwire.com/state-of-security/mitre-framework/the-mitre-attck-framework-lateral-movement/ (visited on 12/07/2018).

[9] J. H. Saltzer and M. D. Schroeder. „The protection of information in computer systems“. In: *Proceedings of the IEEE* 63.9 (Sept. 1975), pp. 1278–1308. ISSN: 0018-9219. DOI: 10.1109/PROC.1975.9939.

[10] *Active Directory Domain Services Overview.* May 31, 2017. URL: https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview (visited on 05/07/2019).

[11] DaniHalfin. *Active Directory Security Groups (Windows 10).* Apr. 19, 2017. URL: https://docs.microsoft.com/en-us/windows/security/identity-protection/access-control/active-directory-security-groups (visited on 04/10/2019).

[12] Wengyik Yeong, Tim Howes, and Steve Kille. *Lightweight Directory Access Protocol.* RFC 1777. RFC Editor, 1995. URL: http://www.rfc-editor.org/rfc/rfc1777.txt.

[13] J. Galvin. *IAB and IESG Selection, Confirmation, and Recall Process: Operation of the Nominating and Recall Committees.* RFC 3777. RFC Editor, 2004. URL: http://www.rfc-editor.org/rfc/rfc3777.txt.

[14] *Group Policy API.* May 31, 2018. URL: https://docs.microsoft.com/en-us/previous-versions/windows/desktop/policy/group-policy-start-page (visited on 05/07/2019).

[15] *NetLocalGroupGetMembers Function (Lmaccess.h).* May 12, 2018. URL: https://docs.microsoft.com/en-us/windows/desktop/api/lmaccess/nf-lmaccess-netlocalgroupgetmembers (visited on 04/09/2019).

[16] *[MS-SAMR]: Security Account Manager (SAM) Remote Protocol (Client-to-Server).* Feb. 14, 2019. URL: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-samr/4df07fab-1bbc-452f-8e92-7853a3c7e380 (visited on 04/09/2019).

[17] Smith Randy Franklin. *Access Denied: Understanding the Anonymous Enumeration Policies.* URL: https://www.itprotoday.com/windows-78/access-denied-understanding-anonymous-enumeration-policies (visited on 04/09/2019).

[18] Justinha. *Security Options (Windows 10).* June 28, 2018. URL: https://docs.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/security-options (visited on 04/09/2019).

[19] Rohan Vazarkar. *SharpHound: Target Selection and API Usage.* Mar. 5, 2018. URL: https://blog.cptjesus.com/posts/sharphoundtargeting (visited on 03/19/2019).

[20] *NetSessionEnum Function (Lmshare.h).* May 12, 2018. URL: https://docs.microsoft.com/en-us/windows/desktop/api/lmshare/nf-lmshare-netsessionenum (visited on 04/09/2019).

[21] *NetWkstaUserEnum Function (Lmwksta.h).* May 12, 2018. URL: `https://docs.microsoft.com/en-us/windows/desktop/api/lmwksta/nf-lmwksta-netwkstauserenum` (visited on 04/09/2019).

[22] *[MS-ADOD]: Glossary.* Feb. 14, 2019. URL: `https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-adod/afa7460b-713c-476d-9af1-5f9a5fe6ab27` (visited on 04/10/2019).

[23] „Chapter 5 - Centrality and Hubs“. In: *Fundamentals of Brain Network Analysis.* Ed. by Alex Fornito, Andrew Zalesky, and Edward T. Bullmore. San Diego: Academic Press, Jan. 1, 2016, pp. 137–161. ISBN: 978-0-12-407908-3. DOI: `10.1016/B978-0-12-407908-3.00005-4`. URL: `http://www.sciencedirect.com/science/article/pii/B9780124079083000054` (visited on 05/07/2019).

[24] John Dunagan, Alice X. Zheng, and Daniel R. Simon. „Heat-ray: Combating Identity Snowball Attacks Using Machine Learning, Combinatorial Optimization and Attack Graphs“. In: *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles.* SOSP ’09. New York, NY, USA: ACM, 2009, pp. 305–320. ISBN: 978-1-60558-752-3. DOI: `10.1145/1629575.1629605`. URL: `http://doi.acm.org/10.1145/1629575.1629605` (visited on 04/25/2018).

[25] Rohan Vazarkar, Will Schroeder, and Andrew Robbins. *BloodHound: Six Degrees of Domain Admin.* BloodHoundAD. URL: `https://github.com/BloodHoundAD/BloodHound` (visited on 07/29/2018).

[26] 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. New York, NY, USA: ACM, 2016, pp. 45–52. ISBN: 978-1-4503-4566-8. DOI: `10.1145/2994475.2994476`. URL: `http://doi.acm.org/10.1145/2994475.2994476` (visited on 11/22/2018).

[27] Rohan Vazarkar, Will Schroeder, and Andrew Robbins. *SharpHound: The BloodHound C# Ingestor.* BloodHoundAD. URL: `https://github.com/BloodHoundAD/SharpHound` (visited on 04/19/2019).

[28] Rohan Vazarkar, Will Schroeder, and Andrew Robbins. *BloodHound-Tools.* BloodHoundAD. URL: `https://github.com/BloodHoundAD/BloodHound-Tools` (visited on 04/19/2019).

[29] Pralhad Chaskar. *Mapping Network Using Sharphound.* June 14, 2018. URL: `https://www.c0d3xpl0it.com/2018/06/mapping-network-using-sharphound.html` (visited on 04/23/2019).

[30] *Lay of the Land with Bloodhound.* URL: `https://threat.tevora.com/lay-of-the-land-with-bloodhound/` (visited on 04/23/2019).

[31] *Defending Windows Domain Against Mimikatz Attacks.* URL: `http://woshub.com/defending-windows-domain-against-mimikatz-attacks/` (visited on 09/24/2018).

[32] *Windows 10 Device Guard and Credential Guard Demystified – Ash's Blog.* URL: `https://blogs.technet.microsoft.com/ash/2016/03/02/windows-10-device-guard-and-credential-guard-demystified/` (visited on 03/17/2019).

[33] Didier Stevens. *Windows Credential Guard & Mimikatz.* URL: `https://blog.nviso.be/2018/01/09/windows-credential-guard-mimikatz/` (visited on 01/26/2019).

[34] Hazaveh Mahdi. *Easily Disable Hyper-V To Run VMWare and Virtual Box.* URL: `https://hazaveh.net/2015/11/easily-disable-hyper-v-to-run-vmware-and-virtual-box/` (visited on 03/17/2019).

[35] DaniHalfin. *Windows Defender Credential Guard Requirements (Windows 10).* Dec. 1, 2018. URL: `https://docs.microsoft.com/en-us/windows/security/identity-protection/credential-guard/credential-guard-requirements` (visited on 03/17/2019).