TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Loc Phan Van 175353IDCR

PROTECTING WINDOWS PRIVILEGED ACCOUNTS

Bachelor's thesis

Supervisor: Ilhan Celebi
MSc. Cyber Security

Tallinn 2018

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Loc Phan Van 175353IDCR

PRIVILEGEERITUD WINDOWSI KONTODE KAITSMINE

Bakalaureusetöö

Juhendaja: Ilhan Celebi
MSc. Küberturvalisus

Tallinn 2018

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Loc Phan Van

17.04.2018

Abstract

Active Directory (AD) is a technology created by Microsoft to provide network services including LDAP directory services, Kerberos based authentication, DNS naming, secure access to resources, and more. Active Directory uses a single Jet database which a variety of services and applications can use to access and store a variety of information. Active Directory is used by system administrators to store information about users, assign security policies, and deploy software [1]. Moreover, AD is being used by many large enterprises and also smaller companies thanks to its benefits. Therefore, there is still a question of how to secure their privileged assets, especially on those privileged objects in AD.

The thesis focuses on finding the best methods to protect the privileged accounts in Windows AD environment, those methods are concluded experiences during the time the author working in Swedbank. In order to accomplish this goal, the thesis points out how important of the privileged accounts impact on the entire organization due to the risks it gives when exploited. Through that, the thesis demonstrates the effectiveness of manipulating the principles as well as implementing some security development by combining built-in resource in Windows and third party open source software. Step by step on how to apply AD delegation principles helps to obtain a good understanding of the AD security infrastructure. Additionally, the thesis also aims to indicate what subjects in AD that needs to be managed, controlled and monitored before giving the development of AD security.

Annotatsioon

Privilegeeritud Windowsi Kontode Kaitsmine

Active Directory (AD) on Microsofti poolt loodud võrguteenuseid toetav tehnoloogia, sealhulgas toetamaks LDAP-i kataloogiteenused; Kerberosel põhinev autentimine, DNS-i nimetamine, turvaline juurdepääs ressurssidele ja palju muud. Active Directory kasutab üht Jet-i andmebaasi, mida mitmesugused teenused ja rakendused saavad kasutada mitmesugustele andmetele ligipääsemiseks ja salvestamiseks. Süsteemiadministraatorid kasutavad Active Directory-i kasutajate andmete salvestamiseks, turvapoliitika määramiseks ja tarkvara juurutamiseks [1]. AD eeliste tõttu kasutavad seda nii suured, kui ka väiksemad ettevõtted. Seepärast on endiselt küsimus, kuidas kindlustada nende privilegeeritud vara turvalisus, eriti nende privilegeeritud objektide puhul AD-s.

Doktoritöö keskendub kontode kaitsmisel parimate optimaalsete lahendusete leidmiseks Windowsi AD keskkonnas. Töös toodud lahendusteni on jõudnud autor oma töö ajal Swedbankis. Töö tulemused näitavad, kui oluline on privilegeeritud kontode mõju tervele organisatsioonile nende ärakasutamisest tekkivate riskide tõttu. Väitekirjas demonstreeritakse põhimõtetega manipuleerimise efektiivsust ning vastu meetmeks turvalisuse tagamiseks kasutusele võetavaid abinõusid, mis on kombineeritud Windowsi enda ning avatud lähtekoodiga tarkvara omavahelisel ühendamisel. Järjestikuste sammudena on välja toodud AD delegeerimise põhimõtete rakendamine, mis aitab saada parema ülevaate turvalisuse tagamise infrastruktuurist. Lisaks eeltoodule on doktoritöö eesmärgiks välja selgitada, milliste aspektidega täpsemalt tuleb riskide maandamisel arvestada ning mida põhjalikumalt uurida enne vastavate meetmete arendamisega alustamist.

List of abbreviations and terms

TUT	Tallinn University of Technology
DC	Domain Controller
RBAC	Role Based Access Control
OU	Organization Unit
PAW	Privileged Access Workstation
GPO	Group Policy
ATA	Advanced Threat Analytics
NIST	National Institute of Standards and Technology
SMTP	Simple Mail Transfer Protocol
HTTP	Hypertext Transfer Protocol
COM	Component Object Model
RD	Remote Desktop
SID	Security Identifier
FSMO	Flexible Single Master Operation
DNS	Domain Name Server
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
SIEM	Security Information and Event Management
LDAP	Lightweight Directory Access Protocol

Table of contents

List of figures

List of tables

1 Introduction

Within the past decades, attack against computing infrastructure, whether simple or complex, have existed as long as internet and computers have; However, increasing numbers of companies of all sizes, in all parts of the worlds have been attacked and compromised in the way that's have significantly changed the threat landscape. Especially, big companies and large enterprise have always become a primary target of attackers for getting most profitable business. Therefore, privileged accounts are exploited every day by advanced and insider attacks to steal billions of dollars' worth of sensitive information. Correspondingly, security frameworks such as the Council on Cyber Security top 20 Critical Security Controls, NIST and others have always maintained the importance of protecting, managing and monitoring privileged accounts. There is wide agreement the security industry that these accounts represent they keys to a company's digital kingdom [2].

Privileged accounts, like regular user accounts, have a valid set of credentials used to gain access to particular system or systems on a given network. The difference, is that privileged accounts credentials provide elevated, non-restrictive access to the underlying platform that non-privileged user accounts don't have access too. These accounts are designed to be used by system administrators to manage or troubleshoot network systems, run services, or allow applications to communicate with one another. The downside is that these same credentials, which are used to help keep business operating can easily be used by attackers or malicious insiders to cause significant damage to the network and organization [2].

The goal of this thesis is to provide drawn from a number of sources and derived from practices designed in order to protect Windows privileged accounts against compromise. Although it is not possible to prevent attacks, it is possible to reduce the Active Directory attack surface and to implement controls that make compromise of the directory much more difficult for attackers and prevent the problems before it becomes larger [3].

1.1 Problem statement and contribution of the thesis

This thesis aims to demonstrate why and how we need to have a proper defence strategy for Windows privileged accounts. Most of the paper completed in this topic had been investigated in Swedbank's infrastructure, thereby it is still problematic to other environments when applied. In other cases, information is scattered between different papers and blog articles. This means there are no recent papers available which would walk through the whole process needed for the best Windows privileged accounts security practices.

This thesis will do by the following:

- Identify the necessity of securing Windows privileged accounts
- Point out the benefits of protecting Windows privileged accounts
- Provide the best security principles recommended by Microsoft
- Provide the best security principles from conclusive experience
- Develop the tools that serve for detecting and mitigating the malicious attacks
- Give out the best practices

The main contribution of this thesis is providing a way how to implement and deploy Windows privileged accounts protection by manipulating from a number of sources and derived from practices designed in Swedbank's training program, in which the production environment contains thousands of sensitive data, as a large enterprise it always exposes to the risk and become a primary target victim of attackers.

In addition to highlight the best practice, the author has also developed some helpful tools for Windows accounts forensic, Windows groups/accounts monitoring and other tools by making a best use of built-in scripting language PowerShell and combining them with another open source program. The purpose of it is not possible to prevent the known attacks, but it will give a tremendously useful overview of how to prevent the issues before it causes larger issues by monitoring them and managing them regularly. Moreover, knowing our assets better will help us to have a better control over the infrastructure.

1.2 Overview of related work

Before going to deeper aspects, the author also implemented some of these solutions and had time to do some investigation in Swedbank. It is worth noting that in order to put those security principles into practice or develop our own security weapons are not straightforwardly. One of the most negative aspects is, for instance, that the large enterprise has heavy data stored in their AD, the number might be calculated by thousands thereby increasing the risk it might arrive. It's extremely dependent on version of the Windows operating system, services and initial infrastructure design, that makes the architect difficult to follow-up the recommended principles or re-design the whole AD environment. It gives explanation of why some big companies tend to pay attention on maintaining and upgrading their own infrastructure as a curial role.

Meanwhile, to manipulate the security development also has its drawback. For example, we desire to have our own security tools as such forensic tool, a monitoring tool that is suitable for our infrastructure. During the implementation, it requires some certain permission for the developer and application to execute them. Thus, it goes against the security principles that are supposed to limit the user rights. For some cases, using the open source software in the wild help us to reach better approach. Surely, we cannot debate its benefits are significant, but we are not able to observe the entire hidden things behind, which means we are gaining risks to our environment.

On the other hand, there is the argument that proprietary programs whose code is closed cannot be as easily inspected by malicious hackers who are looking for security holes to exploit. In other words, closed-source software has the advantage of security by obscurity [4].

Despite the negative effects, to put this action into practice that brings tremendous AD security gains. For example, instead of paying heavy money to production monitoring software we can take advantage of combining between well-known open source software and Windows scripting language, which we can all the way through customize them corresponding to our desires, like we know which parts of our environment are sensitive and needed to monitor, some high privileged groups must be watched as an instance. Obviously, implementing the principles takes tons amount of time to accomplish, especially with thousands of data need to be refreshed. Nevertheless, this is an imperative action for not only large enterprises but also smaller enterprises should be aware of.

1.3 Outline of the thesis

This thesis is organized into five chapters. Chapter 1 gives an introduction to the thesis. Problem statement and previous work on related topics are also discussed. Chapter 2 gives the effectiveness of securing Windows privileged accounts. Chapter 3 aims to provide security principles for Windows AD installation regarding the privileged accounts. Chapter 4 is the main of creative aspect of the thesis that provides security development. Chapter 5 supplies the best practices that help us determine the movements give us the most benefit by maturity level. Finally, chapter 6 concludes the thesis.

1.4 Acknowledgements

I would like to thank my team lead Sebastian Domancich who is team lead of Identity team in Swedbank, has allowed me to involve the infrastructure. I am grateful to my academic supervisors Ilhan Celebi for his patience and guidance throughout the course of this bachelor thesis. The opportunity to work with them as my supervisors exposed me to scientific thinking that is required to become a good researcher.

I am also grateful to the Swedbank identity team for the opportunity to carry out this project. Most importantly, I am grateful to Aleksandr Ilnitšov for his support in providing useful documentations and advises in order to contribute my thesis. I want to thank my lecturer Dr. Kaido Kikkas for reviewing the paper with valuable recommendations that help me to contribute my arguments stronger.

2 Effectiveness of protecting privileged Windows account

Privileged accounts, like regular user accounts, have a valid set of credentials used to gain access to particular system or systems on a given network. The difference, is that privileged account credentials provide elevated, non-restrictive access to the underlying platform that non-privileged user accounts don’t have access too [2].

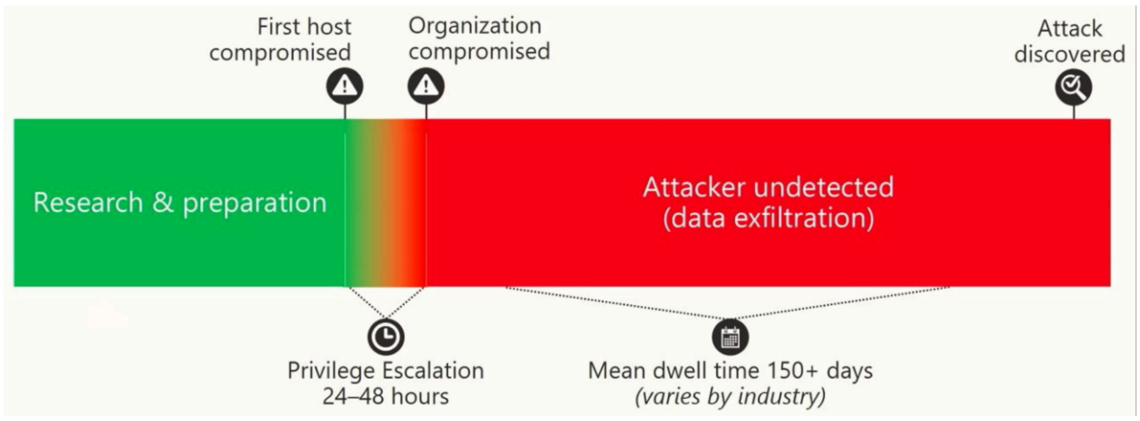

First host compromised
Organization compromised
Attack discovered
Research & preparation
Attacker undetected (data exfiltration)
Privilege Escalation 24–48 hours
Mean dwell time 150+ days (varies by industry)

Figure 1. Timeline for typical attack scenario [5].

The following is a typical attack scenario:

1. The attacker does some research and preparation about an organization (such as by using Facebook, Linked In, search engines, or other social networking services)

2. The attacker determines the best method for initiating an attack (such as a phishing email or probing edge-of-network services)

3. The attacker initiates an attack to gain a foothold into the organization's network and services

4. The attacker gains access and then, using one or more compromised identities, attempts to escalate their privileges

5. The attacker gains escalated privileges and continues to compromise services and servers within the organization, compromising data and/or causing denial of service [5] .

It is important to note that the longer the attacker goes undetected. The more damage they can do and the harder it will be to expunge the attacker from the network. Again, our goal is to extend the time it takes to escalate privilege to weeks and months so that we can detect an attack and respond to it before the attacker can gain full control. The remainder of this guide focuses on how you can make it harder for an attacker to escalate privilege and move freely in the network, and how to detect the attacks sooner [5].

Therefore, in order to prolong the time for an attacker in during the escalation, implementing proper access control is still an important defence. But access controls can be ineffective if poorly implemented. One bad decision can lead to a compromise. You only have to look at many organization's Active Directory (AD) to realize that little thought has been put into how to grant access to the directory, domain controllers (DCs), and other sensitive assets. It is common to find IT support staff with domain administrative privileges, domain admin accounts used to log in to users' PCs, and administrative user accounts and passwords shared across multiple devices [6].

This chapter will provide the effectiveness of manipulating AD security principles and security development against some typical attack types as such malicious attack and abnormal behaviour.

2.1 Extending the time for escalation

As above-mentioned facts, manipulating security principles do not help us to prevent those typical malicious attacks as such Pass-the-Ticket, Pass-the-Hash, Overpass-the-Hash, Golden Ticket and so on. The more time it gives for an attacker to escalate privilege or make a lateral movement freely in the network, the more time we could detect and prevent them before they cause larger issues. Here is an example from one of AD security principle proposed by Microsoft called: Active directory tier model.

Despite the security features Microsoft is including in Windows 10 and Windows Server 2016, implementing proper access controls is still an important defence. The tiered administrative model aims to help organizations to better secure environments. The model defines three tiers that create buffer zones to separate administration of high-risk PCs and valuable assets like domain controllers [6].

Primary responsibilities and critical restrictions

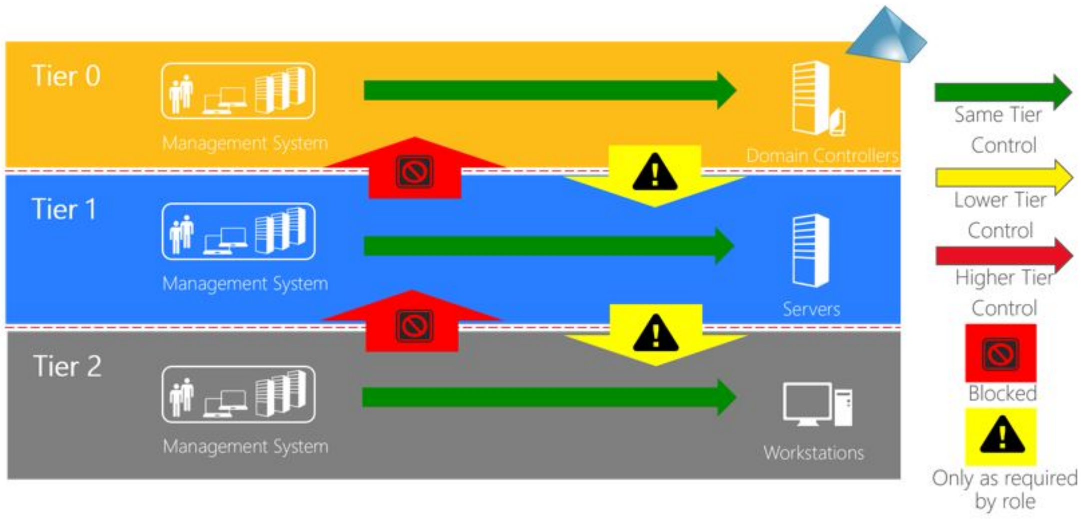


Figure 2. Active Directory Tiered Administrative Model Control Restrictions [6].

Tier 0 administrator - manage the identity store and a small number of systems that are in effective control of it, and:

- Can manage and control assets at any level as required
- Can only log on interactively or access assets trusted at the Tier 0 level [7].

Tier 1 administrator - manage enterprise servers, services, and applications, and:

- Can only manage and control assets at the Tier 1 or Tier 2 level
- Can only access assets (via network logon type) that are trusted at the Tier 1 or Tier 0 levels
- Can only interactively log on to assets trusted at the Tier 1 level [7].

Tier 2 administrator - manage enterprise desktops, laptops, printers, and other user devices, and [7]:

- Can only manage and control assets at the Tier 2 level
- Can access assets (via network logon type) at any level as required
- Can only interactively log on to assets trusted at Tier 2 level [7].

Logon restrictions

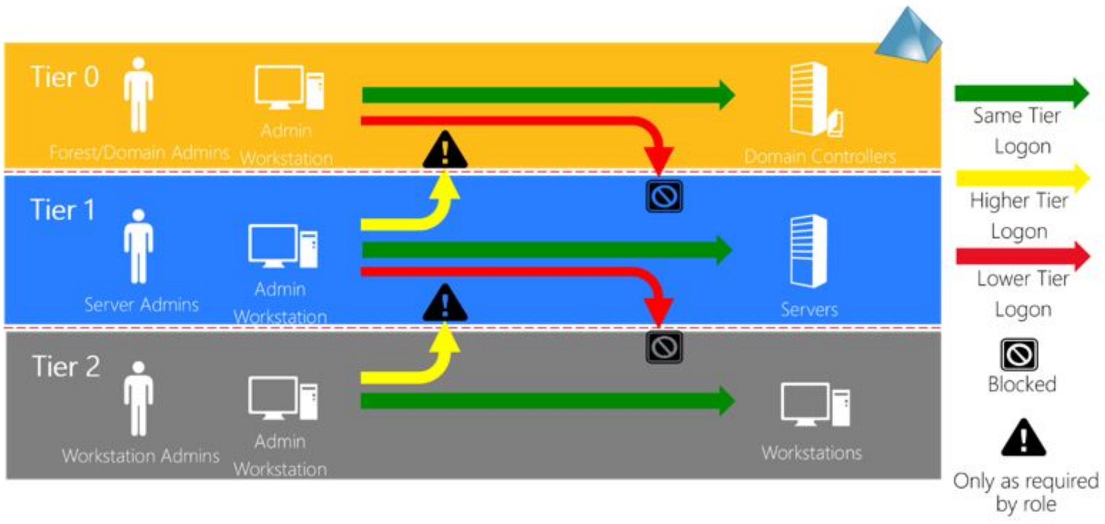


Figure 3. Active Directory Tiered Administrative Model Control Restrictions [7].

The three tiers increase the cost for an attacker trying to compromise sensitive systems. You should consider that a user that has full access to all Tier 2 assets could get access to assets in a higher tier. The tiered administrative model makes it harder for a hacker to move from a Tier 2 to a Tier 0 asset but doesn't make it impossible. The tiers also serve as a basic prioritization mechanism for protecting administrative assets, but it is important to consider that an attacker with control of all assets at any tier can access most or all business assets [6].

In addition, the reason it is useful as a basic prioritization mechanism is attacker difficulty/cost. It is easier for an attacker to operate with full control of all identities (Tier 0) or servers and cloud services (Tier 1) than it is if they must access each individual workstation or user device (Tier 2) to get your organization's data [7].

In conclusion, all described above have mentioned the fact that applying principle recommended by provider here is Microsoft, it gives the reliability in mitigating those malicious attacks. The time for an attacker trying to escalate privilege will dramatically increase, thanks to that the administrator or security engineer have more time to detect regardless of they stay there for years but cannot affect to the whole system.

2.2 Detecting abnormal behaviour

For cybercriminals, credentials that provide access to corporate networks are invaluable. Phishing scams, social engineering, and malware are just a few of the popular techniques by which these criminals acquire employee corporate credentials. These techniques are effective – Verizon reports that over 80% of advanced attacks are due to external actors [8].

Time until discover	Percentage
More than a year	5.9
About a year	10.3
Several months	16.6
Several weeks	18.3
Several days	22.9
Within a day	12.3
Within few hours	8.7
Almost instantly	3.6

Table 1. Security incident discovery by IT Security Risks Survey 2017 [9].

Once attackers have these credentials, they can pose as users with legitimate access, scour internal networks for valuable data (e.g., emails, intellectual property, financial information, etc.) and steal it with impunity overextended periods of time without raising any alarms. In fact, over 75% of attacks take weeks or more to be detected. Traditional rule-based systems such as IDS/IPS or SIEM are good at detecting the "known bad" attacks, but are ill-equipped to detect such credential-based attacks that fall in the realm of the "unknown bad" [8].

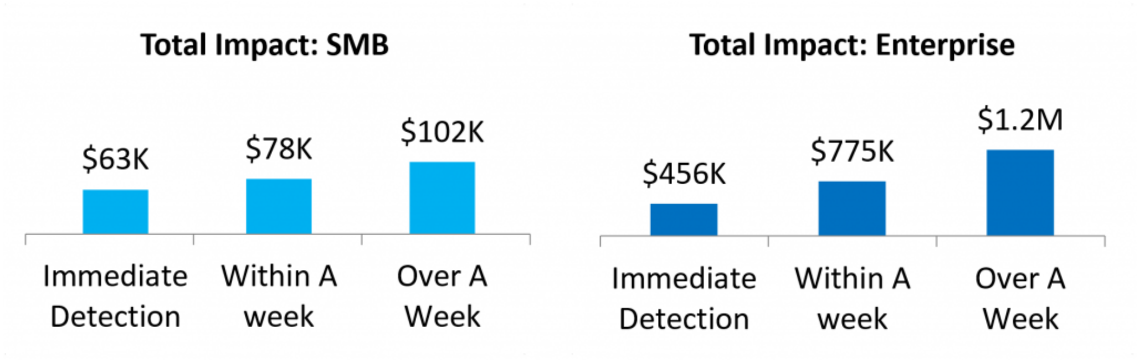
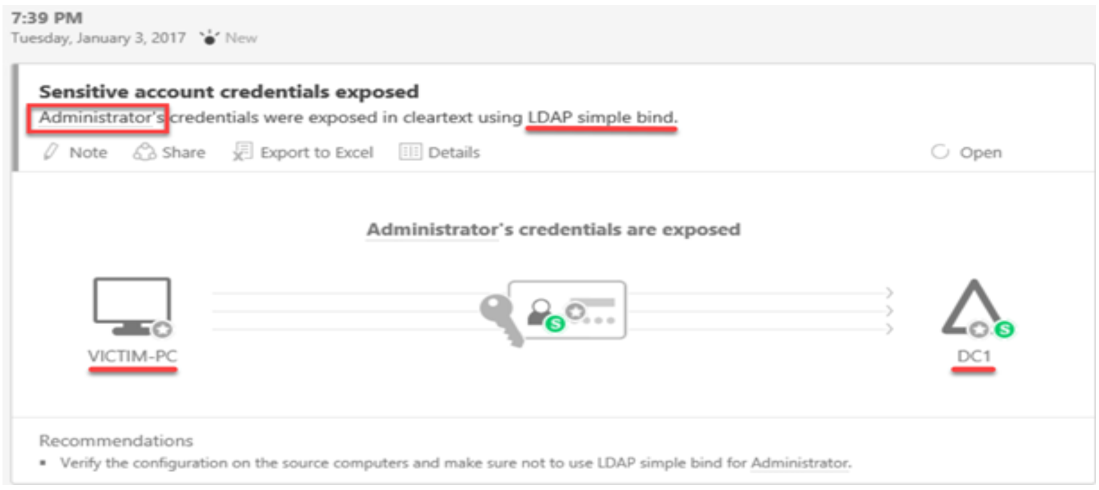


Figure 4. the average cost of recovering from a targeted attack [9].

Nevertheless, detecting abnormal behaviour of Windows privileged accounts are not strenuous. Attackers add users to highly privileged groups. They do so to gain access to more resources and to gain persistency. The detection relies on profiling the group modification activities of users and alerting when an abnormal addition to a sensitive group is seen. There are many well-known Windows built-in software as such Advanced Threat Analytics (ATA) [10].

ATA is deployed it begins monitoring the activity of all the entities in the organization, learning the normal behavior of entities, and detecting abnormal behavior and known techniques used by advanced attackers and insiders [11].


7:39 PM
Tuesday, January 3, 2017 New

Sensitive account credentials exposed

Administrator's credentials were exposed in cleartext using LDAP simple bind.

Recommendations
- Verify the configuration on the source computers and make sure not to use LDAP simple bind for Administrator.

Figure 5. ATA detects sensitive account credentials exposed [12].

Abnormal behavior is detected by ATA using behavioural analytics and leveraging Machine Learning to uncover questionable activities and abnormal behavior in users and devices in your network, including:

- Anomalous logins
- Unknown threats
- Password sharing
- Lateral movement
- **Modification of sensitive groups** [10].

What we should mention here is modification of sensitive groups which grants an account to have more privileges in environment for instances:

- **Administrators:** Local or Active Directory group. The AD group has full admin rights to the Active Directory domain and Domain Controllers

  Members of the Administrators group have complete and unrestricted access to the computer, or if the computer is promoted to a domain controller, members have unrestricted access to the domain

- **Domain Admins:** Active Directory group with full admin rights to the Active Directory domain and all computers (default), including all workstations, servers, and Domain Controllers. Gains this right through automatic membership in the Administrators group for the domain as well as all computers when they are joined to the domain

Members of the Domain Admins security group are authorized to administer the domain. By default, the Domain Admins group is a member of the Administrators group on all computers that have joined a domain, including the domain controllers. The Domain Admins group is the default owner of any object that is created in Active Directory for the domain by any member of the group [13].

At chapter 4, the thesis will describe more detail about the desire of taking the advantage of combining between Windows built-in scripting language PowerShell and Zabbix monitoring open source software. With this combination, you can easily build your own weapon that can be customized and configured corresponding your needs as such we can contribute more groups to monitor, which types of alerts, notifications you want to achieve. Furthermore, you are also able to create the tool for keeping track on accounts behaviours, thanks to that it helps to give a most accurate assessment.

3 Windows AD security principles

This chapter describes how we should adhere to the principles that are concluded from realistic practices and highly recommended by supplier Microsoft. It also points out the benefits it gives, in making a good security strategy for your Windows AD environment, especially in securing privileged accounts. Those principles for delegation is the main aspect that will be focused in this chapter.

The delegation capabilities in Active Directory are quite powerful, addressing a number of security issues and simplifying management tasks. By properly delegating rights within Active Directory, you can enforce specified roles in the environment, limit the impact and likelihood of administrative error, and apply the principle of least privilege throughout your infrastructure. Yet many of the organizations that rely on Active Directory have yet to tap into the power of delegation. In part, this is because, on the surface, developing an Active Directory delegation model for your enterprise appears to be fairly complex. While the largest hurdle is to develop a delegation model that fits the unique needs of your organization, the truth is that there are very simple models that can be applied to most IT infrastructures with little modification [14].

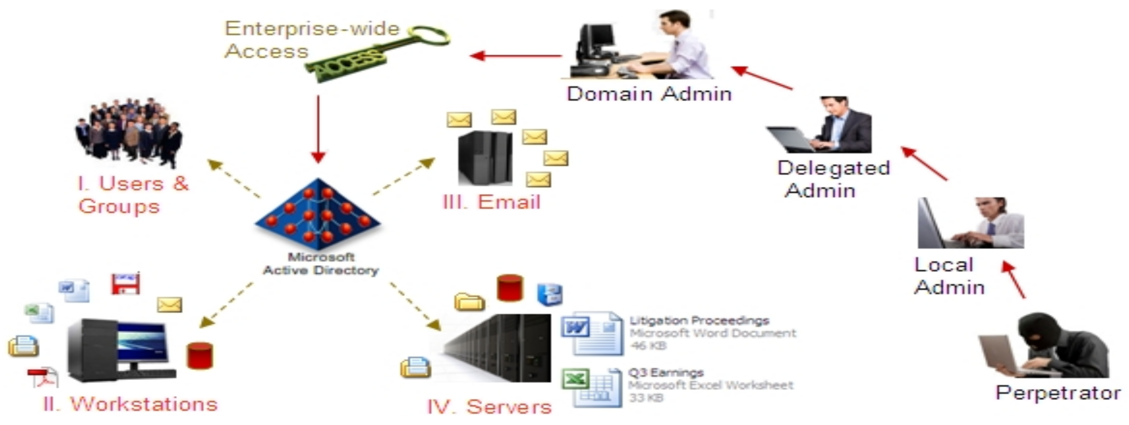


Figure 6. Windows escalation process [15].

Although every environment is different in some way, shape, or form, the reality is that most large enterprises are similar in many ways and face the same IT challenges. For instance, many organizations are divided into geographic regions, have evolved from separate IT engineering or operational support teams, and have independent business units. And many large organizations must deal with such matters as privilege escalation as figure 6 has described the step by step how attackers trying to leverage their permission, service account abuse, and "trust." [14].

Meanwhile, for any Active Directory implementation, the administrators must define the rules of engagement for applications that utilize the Active Directory infrastructure. Unfortunately, a common approach-one that is often cited in application installation guides-is to make a service account a member of the Domain Admins group [14].

While these hurdles may seem insurmountable, they represent a prime scenario for implementing an Active Directory delegation model [14]. Developing a delegation model is an iterative design process and the thesis suggests you follow these steps:

- Role based access control
- Tier model
- Dealing with tier-0 groups
- Secondary account
- Privileged admin workstation
- Naming convention [16].
- Privileged accounts cleaning-up

As the chapter 2 already described the effectiveness of cohering these principles will help to extend the time of hacker trying to escalate privilege account or even give them more challenge. Apparently, the administrators and security engineers have more time to detect their movement as well as to have a good precaution before it gets serious. Before going through the principles, the author wants to introduce how to detect the environment in which is being suffered from privileged accounts vulnerability.

3.1 Detecting environmental vulnerability to privileged accounts

In order to detect the AD environment that might expose to the risk. This usually happens in smaller enterprises or companies where are not many high privileged accounts or regular user accounts stored. Therefore, they tend to ignore the principles instead of following the rules which help to prevent and mitigate the risk due to the high amount of money it takes and lazy moment of administrators and security engineers. With larger enterprises, this usually occurs when the administrators leave and related documents are not updated, even no documents have been created. It’s linked to the fact that the new person is not clear what is involved and they intend for the environment not up-to-date. Here are few examples to detect bad AD design that opens up the opportunity for attackers to exploit.

Administrative model: AD facilitates delegation of administration as such rights, permissions, users are able to read most information, additional privileges can be granted in built-in “privileged” groups. But they do not limit the number of privileged user accounts, make user accounts more sensitive for an attacker by giving them a variety of rights to modify the assets.

Naming convention: Usually, this is a crucial rule that needs to be followed-up by Windows administrator and Windows security engineer. In order to avoid the mess in environment that leads to hard control over the system, the setup assets (groups, accounts) naming convention also assist the administrator to detect the suspicious behaviour on-going through the environment.

Lack of monitoring: Surely, most of the company will spend their money for logging and monitoring solution. Nevertheless, there are still many of them do not pay attention on it. Due to the high amount of money it costs in Windows environment, they are not willing to pay for an expensive monitoring software. Thus, they cannot handle the problems before it becomes more serious angles such as an account has been added to administrative groups by unknown user.

Redundant data: Some large enterprises tend to keep the data of employees after they leave for years. It incidentally helps attacker hide their backdoor easily and make administrator harder to have a good control over the properties.

3.2 Role based access control

Role-based access control (RBAC) restricts network access based on a person's role within an organization and has become one of the main methods for advanced access control. The roles in RBAC refer to the levels of access that employees have to the network [17].

Employees are only allowed to access the information necessary to effectively perform their job duties. Access can be based on several factors, such as authority, responsibility, and job competency. In addition, access to computer resources can be limited to specific tasks such as the ability to view, create or modify a file [17].

RBAC is also considered as an evolution of the notion of group based permission in the file systems. RBAC contains 04 components:

- Users
- Roles
- Permissions
- Objects

RBAC	AD Delegation Model
User	Privileged account
Roles	Role Group
	Task Group
Permission	Access Control Entry/User Privileged/Logon right
Objects	OU/ Security Policy Setting

Table 2. RBAC and AD Delegation Model [16].

As a result, lower-level employees usually do not have access to sensitive data if they do not need it to fulfil their responsibilities. This is especially helpful if you have many employees and use third-parties and contractors that make it difficult to closely monitor network access. Using RBAC will help in securing your company's sensitive data and important applications [17].

Concept:

Assume that we have 4 factors:

- Privileged accounts
- Roles
- Takes
- Resources/ Targets [16]**:**

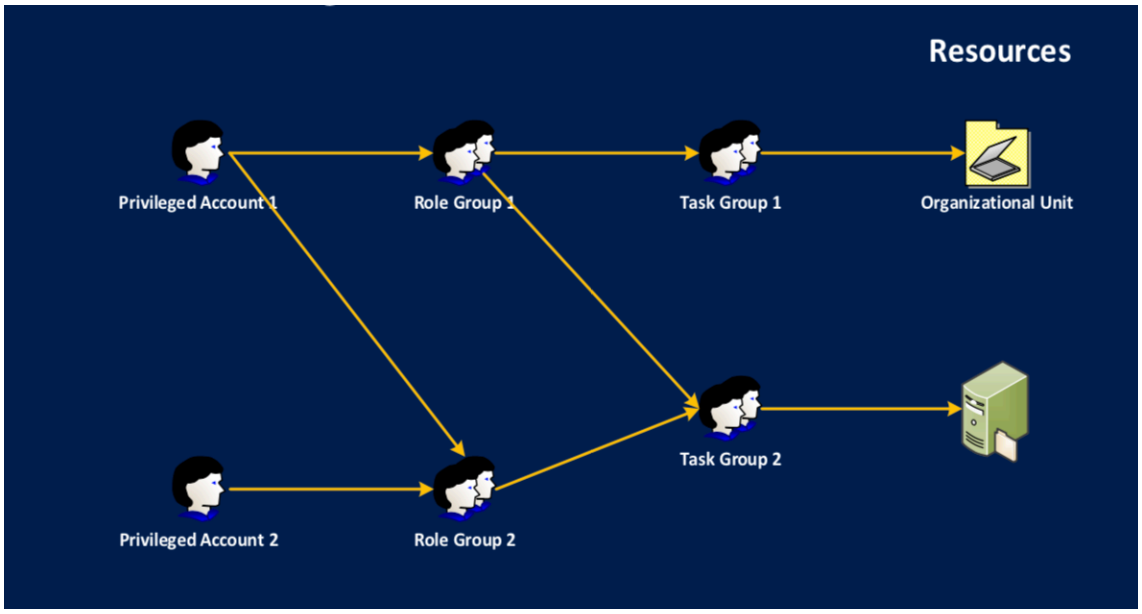


Figure 7. RBAC first concept [16].

In this scenario described in figure 8, we can easily observe that accounts can be member of multiples roles, privileged accounts 1 can have 2 roles in Role Group 1 and Role Group 2 – One to many (1-M), which means privileged accounts 1 becomes a secondary account.

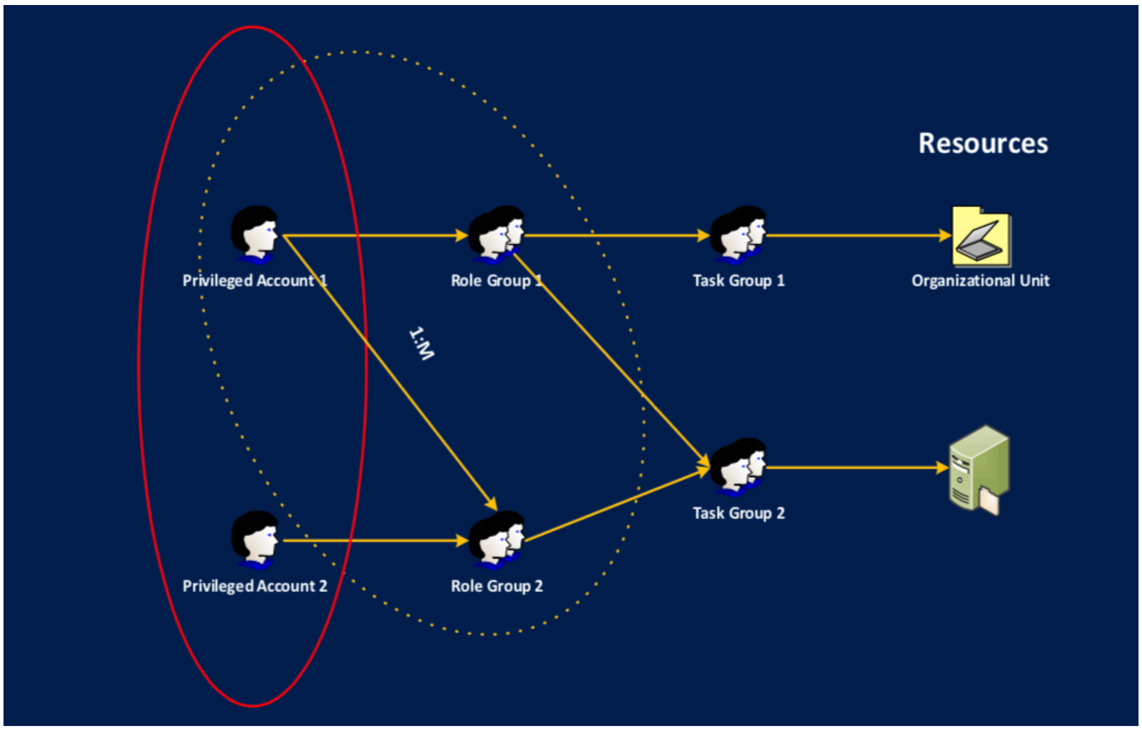


Figure 8. RBAC privileged accounts [16].

The roles here are represented by a security group. It's also granted for multiple tasks groups 1 and 2 – One to Many (1-M). The role represents a job function as figure 9.

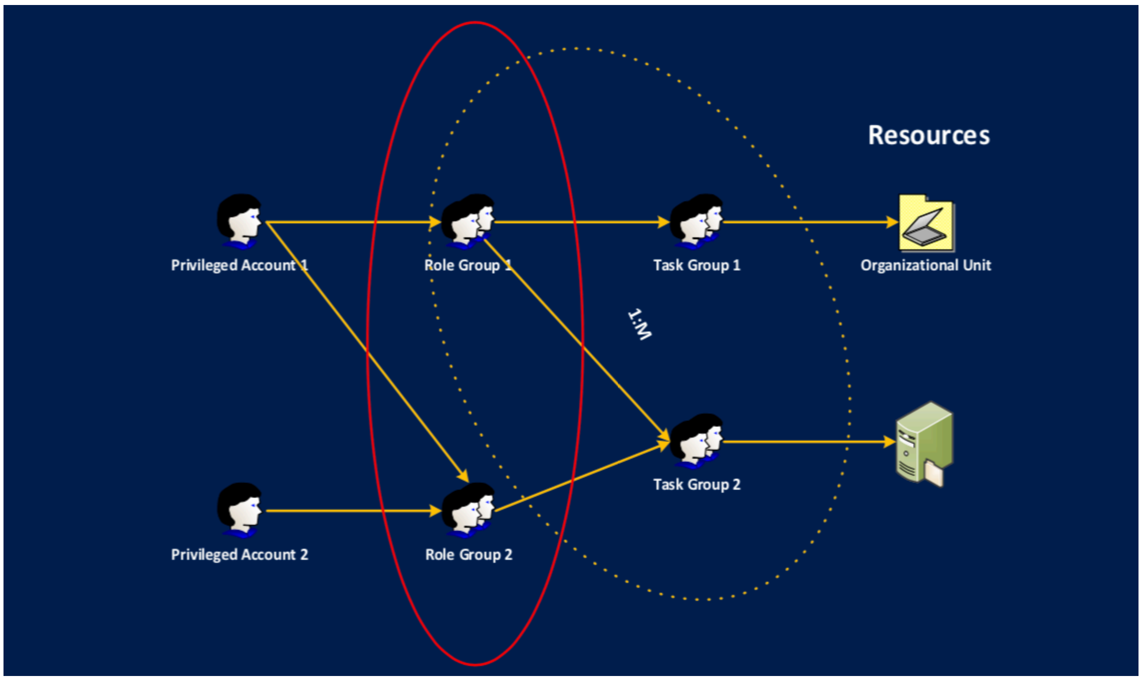


Figure 9. RBAC Role Definition [16].

Illustrated in figure 10, tasks are actions carried out in AD, at this far, administrators should grant the permission for task. More permission it has, the more we have to deal with undesired risks. Therefore, the least privileged should be used in this case.

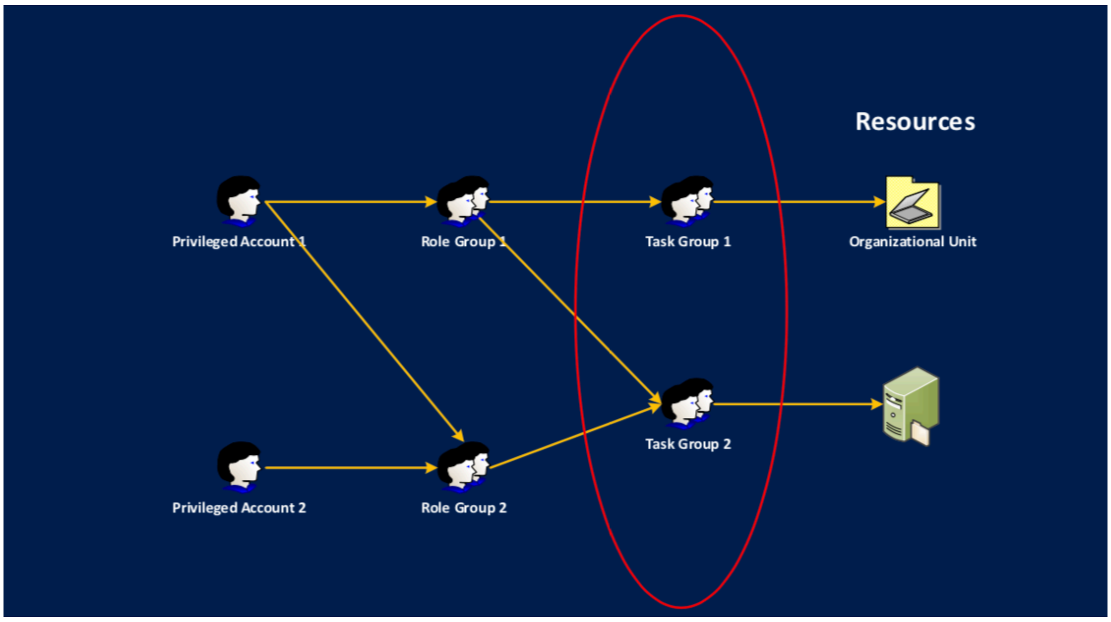


Figure 10. RBAC Task Definition [16].

Resources could be any targets in AD, for examples: An Organization Unit, User Right/ Logon Right, Shared Folder / File System and so on.

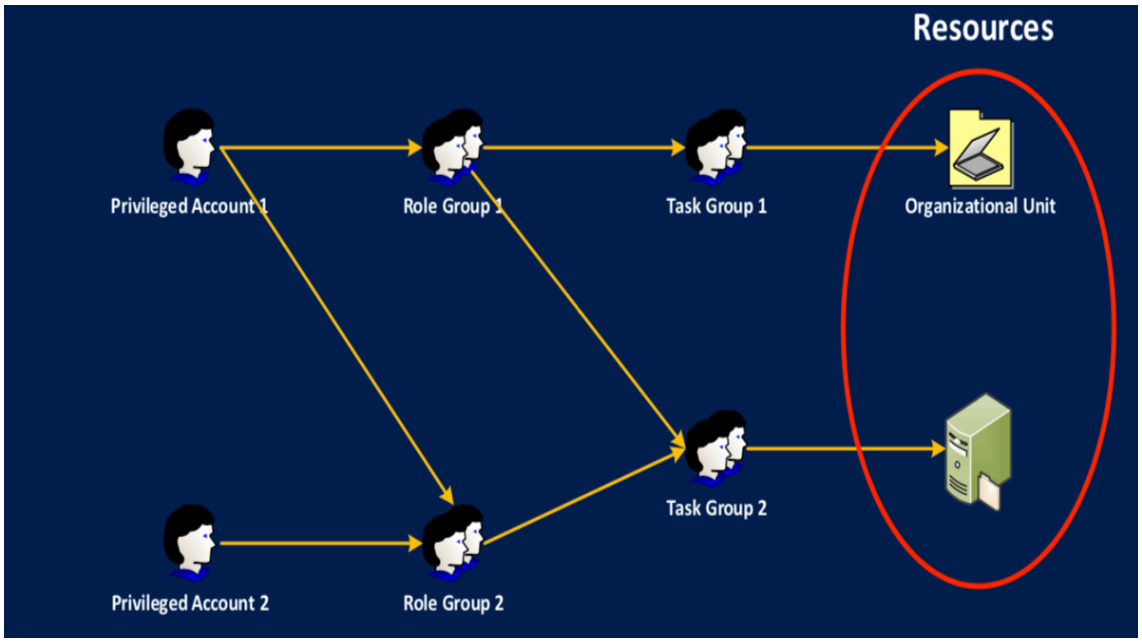


Figure 11. RBAC Resource Definition [16].

Thanks to RBAC, you can handle what end-users can do at both broad and granular levels. You can indicate whether the user is an administrator, a specialist user, or an end-user, and align roles and access permissions with your employees' positions in the organization. Permissions are allocated only with enough access as needed for employees to do their jobs [17].

Moreover, by adding a user to a role group, the user has access to all the roles in that group. If they are removed, access becomes restricted. Users may also be assigned to multiple groups in the event they need temporary access to certain data or programs and then removed once the project is complete. Other options for user access may include:

- Primary – the primary contact for a specific account or role
- Billing – access for one end-user to the billing account
- Technical – assigned to users that perform technical tasks
- Administrative – access for users that perform administrative tasks [17].

Advantages of RBAC

In terms of information security, managing and auditing network access is extremely essential. Access can and should be granted on a need-to-know basis. Especially, in larger enterprise like Swedbank as an example, with thousands of employees, security is more easily maintained by limiting unnecessary access to sensitive information based on each user's established role within the organization. Other advantages include:

- **Reducing administrative work and IT support:** Thanks to RBAC, you can reduce the need for paperwork and password changes when an employee is hired or changes their role
- **Maximizing operational efficiency:** Instead of trying to administer lower-level access control, all the roles can be aligned with the organizational structure of the business and users can do their jobs more efficiently and autonomously
- **Improving compliance:** With an RBAC system in place, companies can more easily meet statutory and regulatory requirements for privacy and confidentiality as IT departments and executives have the ability to manage how data is being

accessed and used. This is especially significant for health care and financial institutions, which manage lots of sensitive data [17].

3.3 Tier model

This model has already slightly described in chapter 2 showing *"effectiveness of manipulating Tier model"*. This section will represent more detail of how to apply this model as an AD administrative model as well as to indicate the benefits it grants to you. The purpose of this tier model is to protect identity systems using a set of buffer zones between full control of the Environment (Tier 0) and the high-risk workstation assets that attackers frequently compromise [7].

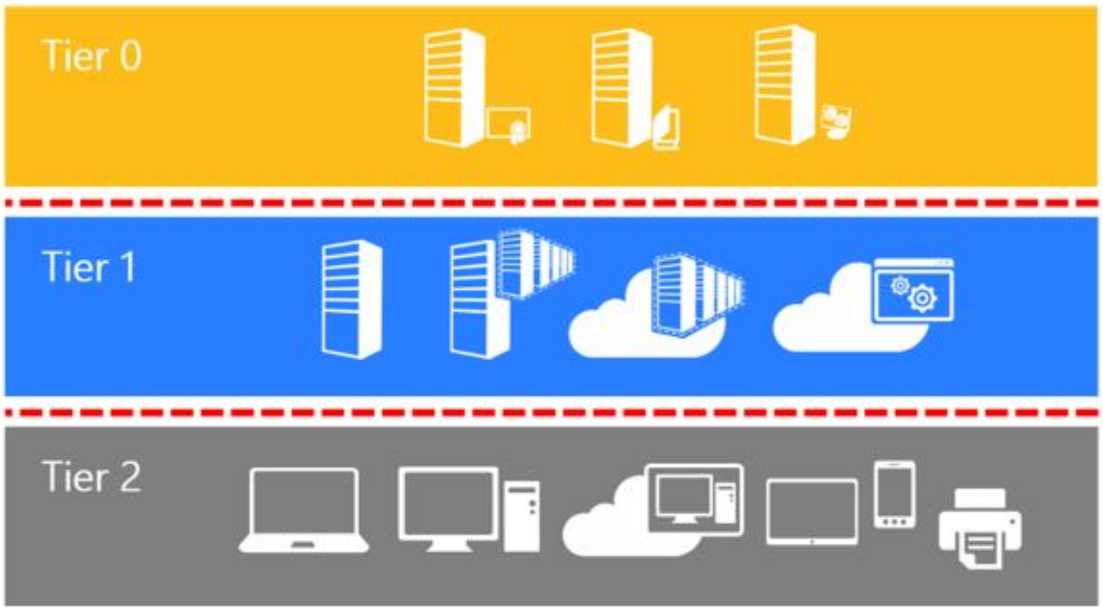


Figure 12. Security Segmentation Model [7].

Tier 0 - Forest admins: Direct or indirect administrative control of the Active Directory forest, domains, or domain controllers
Tier 1- Server admins: Direct or indirect administrative control over a single or multiple servers
Tier 2 - Workstation Admins: Direct or indirect administrative control over a single or multiple devices

Table 3. Tiers definition [16].

The Tier model is composed of three levels and only includes administrative accounts, not standard user accounts:

- **Tier 0** - Direct Control of enterprise identities in the environment. Tier 0 includes accounts, groups, and other assets that have direct or indirect administrative control of the Active Directory forest, domains, or domain controllers, and all the assets in it. The security sensitivity of all Tier 0 assets is equivalent as they are all effectively in control of each other
- **Tier 1** - Control of enterprise servers and applications. Tier 1 assets include server operating systems, cloud services, and enterprise applications. Tier 1 administrator accounts have administrative control of a significant amount of business value that is hosted on these assets. A common example role is server administrators who maintain these operating systems with the ability to impact all enterprise services
- **Tier 2** - Control of user workstations and devices. Tier 2 administrator accounts have administrative control of a significant amount of business value that is hosted on user workstations and devices. Examples include Help Desk and computer support administrators because they can impact the integrity of almost any user data [7].

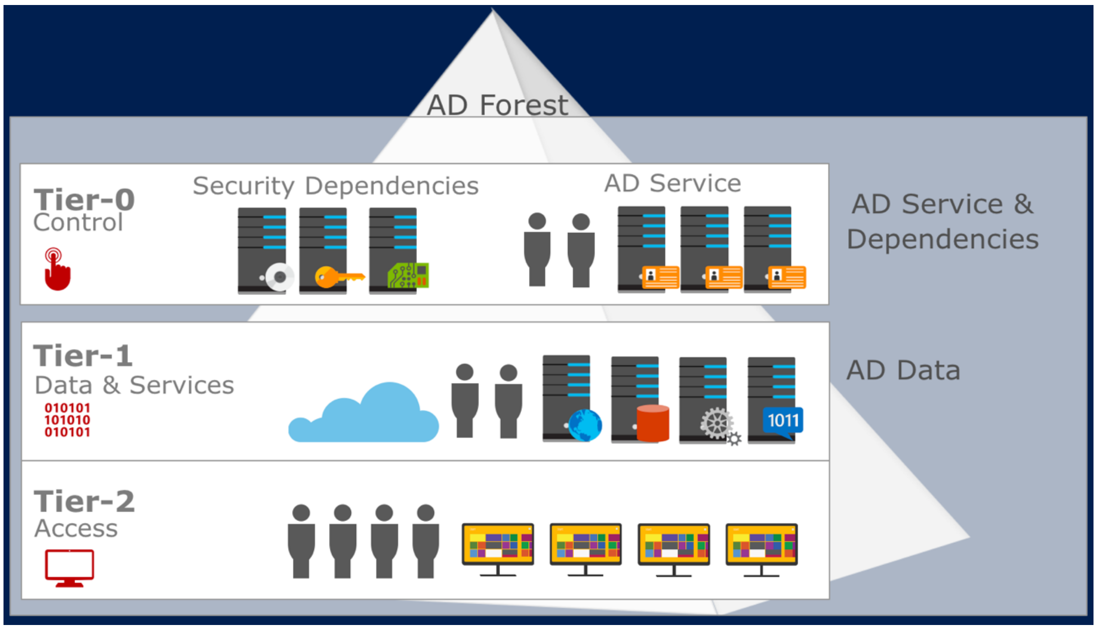


Figure 13. Tier Model [16]

The figure 13 illustrates the AD structure applying Tier model where every tier contains its own inheritances, where Tier 0 administrator - manage the identity store and a small number of systems that are in effective control of it, and it can manage and control assets at any level as required and only log on interactively or access assets trusted at the Tier 0 level [7].

Tier 1 administrator - manage enterprise servers, services, and applications, and it can only manage and control assets at the Tier 1 or Tier 2 level and only access assets (via network logon type) that are trusted at the Tier 1 or Tier 0 levels. It also can only interactively log on to assets trusted at the Tier 1 level [7].

Tier 2 administrator - manage enterprise desktops, laptops, printers, and other user devices, and it can only manage and control assets at the Tier 2 level, access assets (via network logon type) at any level as required and it can only interactively log on to assets trusted at Tier 2 level [7].

For Built-in Tier-0 Groups

→Only use Built-in groups sparingly and only for managing tier-0 [16].

For resource management

→ Delegate privileges over target resources only [16].

For tier management

→ If tier management is needed, an account can manage one tier at a time only [16].

For service accounts

→ If possible, replace domain accounts with a local service identities.

→ Otherwise follow the least privilege and the tier model [16].

The tiers also serve as a basic prioritization mechanism for protecting administrative assets, but it is important to consider that an attacker with control of all assets at any tier can access most or all business assets. The reason it is useful as a basic prioritization mechanism is attacker difficulty/cost. It is easier for an attacker to operate with full control of all identities (Tier 0) or servers and cloud services (Tier 1) than it is if they must access each individual workstation or user device (Tier 2) to get your organization's data [7].

Benefits of Tier Model

Understanding the tiered model gives you a better insight into Microsoft’s security best practices. For example, a Privileged Access Workstation (PAW) that is used by a domain administrator is also considered a Tier 0 asset. A Tier 0 administrator must use a Tier 0 PAW to manage other Tier 0 assets, such as domain controllers because the account will be a member of a highly-privileged domain or forest group [6].

The tiered administrative model isn’t hard to implement. It does require additional resources, like PAWs, and some planning in how to manage access and control between the tiers. It is achievable for most organizations and goes a long way to implementing effective access controls that will keep hackers from compromising sensitive systems [6].

3.4 Dealing with tier-0 groups

The challenge is often determining what access each group actually has. Often the full impact of what access a group actually has is not fully understood by the organization. Attackers leverage access (though not always privileged access) to compromise Active Directory [13].

The key point often missed is that rights to Active Directory and key resources is more than just group membership, it is the combined rights the user has which is made up of:

- Active Directory group membership
- AD groups with privileged rights on computers
- Delegated rights to AD objects by modifying the default permissions (for security principals, both direct and indirect)
- Rights assigned to SIDs in SIDHistory to AD objects
- Delegated rights to Group Policy Objects

- User Rights Assignments configured on workstations, servers, and Domain Controllers via Group Policy (or Local Policy) defines elevated rights and permissions on these systems
- Local group membership on a computer or computers (similar to GPO assigned settings)
- Delegated rights to shared folders [13].

Tier-0 groups:

Domain Admins	Active Directory group with full admin rights to the Active Directory domain and all computers (default).
Enterprise Admins	Active Directory group with full admin rights to all Active Directory domains in the AD forest and gains this right through automatic membership in the Administrators group in every domain in the forest.
Schema Admins	Members of the Schema Admins group can modify the Active Directory schema. This group exists only in the root domain of an Active Directory forest of domains.
Backup Operators	Local or Active Directory group. AD group members can backup or restore Active Directory and have logon rights to Domain Controllers (default).
Server Operators	Members in the Server Operators group can administer domain servers. This group exists only on domain controllers. By default, the group has no members. Members of the Server Operators group can sign in to a server interactively, create and delete network shared resources, start and stop services, back-up and restore files, format the hard disk drive of the computer, and shut down the computer. This group cannot be renamed, deleted, or moved.

Print Operators	Members of this group can manage, create, share, and delete printers that are connected to domain controllers in the domain. They can also manage Active Directory printer objects in the domain. Members of this group can locally sign in to and shut down domain controllers in the domain.
Account Operators	Active Directory group with default privileged rights on domain users and groups, plus the ability to logon to Domain Controllers.
Administrators	Local or Active Directory group. The AD group has full admin rights to the Active Directory domain and Domain Controllers.

Table 4. Tier-0 Groups [13].

Active Directory is intended to facilitate delegation of administration and the principle of least privilege in assigning rights and permissions. "Regular" users who have accounts in an Active Directory domain are, by default, able to read much of what is stored in the directory, but are able to change only a very limited set of data in the directory. Users who require additional privilege can be granted membership in various privileged groups that are built into the directory so that they may perform specific tasks related to their roles, but cannot perform tasks that are not relevant to their duties [18].

3.5 Secondary account

The key to a successful delegation model is enforcing the principle of least privilege. In practice, this means that a security principal should have only the ability to perform the tasks required for its given role and nothing more. Unfortunately, many IT administrators use the same security principal for directory administration and for everyday tasks like Web browsing and reading e-mail. Having separate accounts decreases the likelihood that a tiered administrator will damage the directory service by mistake or fall victim to an attack that is delivered via everyday applications but targeted at the directory administrator [14].

For an example, if administrator just wants to use one account for daily tasks such as browsing, reading email and so on. This is extremely dangerous for him/her and for organization in general. Because the attackers can abuse this habit to deploy an attack like email fishing or he/she could download some malicious files from the internet under administrative account. Therefore, this is not highly recommended to perform as an administrative account daily tasks.

To accomplish this without requiring the user to log off and back on, you use the Secondary Logon service (Runas.exe). This allows users to elevate their privileges by providing an alternate set of credentials when executing scripts or executables on servers and workstations [14].

Solution:

1. One account for normal user activities:
    - Mail
    - Internet Browsing
    - Line of business applications
2. Separate account for doing administration.
3. One admin account for each tier [16].

Example:

A person who is managing AD would have:

- One for daily work like reading emails etc.
- One for managing Tier 1 resources like objects in AD
- One for managing Tier 0 resources like domain controllers [16].

While the concept of using least-privileged accounts is relatively simple, organizations sometimes find it hard to enforce as old IT habits may be rather difficult to break. A straightforward approach to preventing the use of privileged accounts for everyday tasks is to not provide mail access to these accounts in Exchange Server, enforcing this through administrative policy within the organization. This relatively simple approach significantly reduces the likelihood that such accounts will be used for routine, non-administrative tasks [14].

3.6 Privileged admin workstation

Privileged Access Workstations (PAWs) provide a dedicated operating system for sensitive tasks that is protected from Internet attacks and threat vectors. Separating these sensitive tasks and accounts from the daily use workstations and devices provides very strong protection from phishing attacks, application and OS vulnerabilities, various impersonation attacks, and credential theft attacks such as keystroke logging, Pass-the-Hash, and Pass-The-Ticket [19].

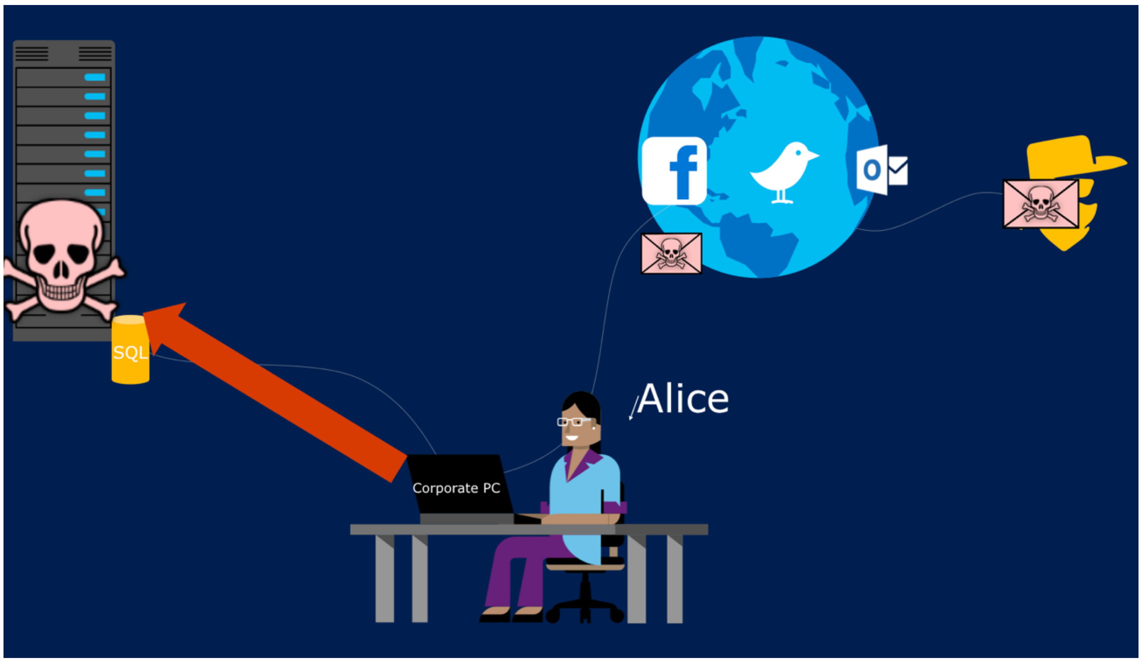


Figure 14. Threats of using PAW [16].

Described by figure 14, the admin workstation Alice is using an admin account for usual activities which are connected to the outside of the organization such as browsing and reading email. Alice does not know that she is exposed to risks accidentally. From browsing some unsecure website, she could download unknown files that attached malicious code and it's literally created by attackers in order to trick the end-users execute it indirectly. Simultaneously, Alice also used to use her administrative account to perform her critical tasks such as to login into SQL server in which contains thousands of user credentials of the organization.

This PAW part is intended to help you implement this capability for protecting high value accounts such as high-privileged IT administrators and high sensitivity business accounts:

- Privileged creds not exposed on standard clients
- Credential exposure contained to the same trust level
- Enables tighter security controls
- Restrict exposure of credentials to only trusted hosts
- Provide a high-security workstation to administrators so they can easily perform administrative tasks [14] [16].

Additionally, we should also have a good rule in order to harden and restrict the sensitive accounts as such:

- Secure Boot to mitigate against attackers or malware
- Software restriction
- Full volume encryption
- USB restrictions
- Network isolation
- Host firewalls
- Antimalware
- Attack surface analysis [16].

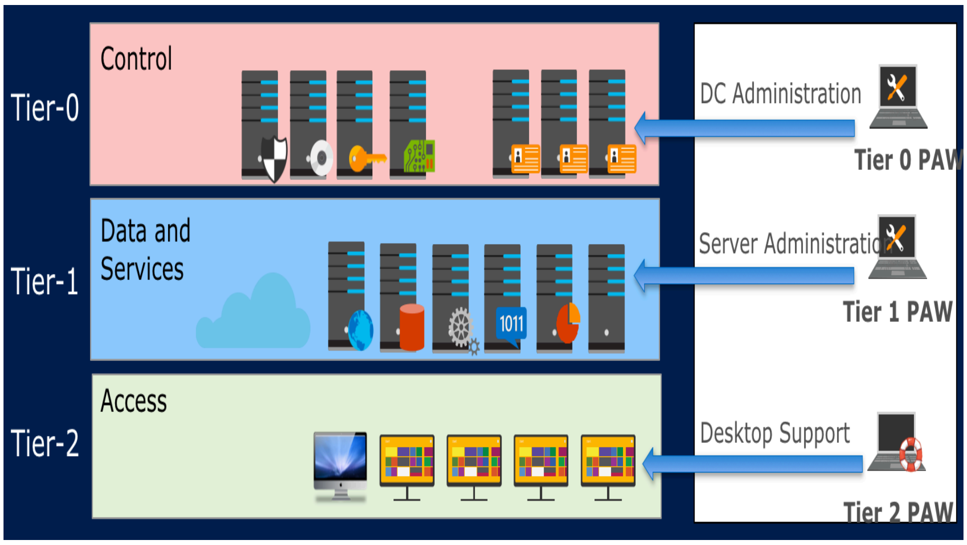


Figure 15. Harden and restrict PAWs in Tier Model [16].

As figure 15 describes, restricting the sensitive accounts to using only hardened PAWs is a straightforward protection for these accounts that is both highly usable for administrators and very difficult for an adversary to defeat [19].

3.7 Naming convention

Implement standard naming conventions across your organization to make identifying critical information about a group much easier. Group names can include critical details about the group, such as the level of access, type of resource, level of security, group scope, mail capability, etc. In order to separate between roles and tasks, roles in different tiers [20]:

Role:

Example:

- Role-<Role Tier>-<Role Name>
- Prefix: Role
- Role Tier: T0/T1/2
- Role Name: Job Function [16]:

Example:

- Role-T2-WorkstationAdmins [16].

Task:

Example:

- Task-<Target Object Type>-<Operation>-<Target>
- Prefix: Task
- Target Object Type: AD object type
- Operation: Create/Delete/Manage etc.
- Target: Short Name for OU/Target [16].

Example:

- Task-Computer-Create-CORP [16].

3.8 Privileged accounts cleaning-up

In order to keep assets clean and secured, many of large enterprises they used to forget about keeping track on their objects because they rely on their security officers or administrators who are responsible for the entry. When the author came to do some investigation on how clean of their AD inheritance, the author realized two things are still problematic and not yet discovered.

First of all, the objects (accounts, groups, etc.) created for long time ago are not controlled and manged. Due to some common reasons such as the AD architect left the company, and the handover was interrupted or not done successful. Therefore, the newcomers who are going to manage this task would not have a capability to obtain what involved and the compresence about his/her infrastructure, it’s linked to the fact that all the resources are being kept not completely secured and watched by the stakeholders. What would happen if the accounts had been created 10 years ago and not being logged on for last 7 years, and importantly these are still available on the AD database. Obviously, those accounts would be abused by the attackers to use as an authenticated user with high level of trust, it helps the attackers to reside on the organization for years without being detected.

Secondly, after realizing this problem is exposed to the risk and it needs to be dealt as soon as possible. Having a good assessment on the data but there is still a question for them that what should they do next? Due to the convenience of employee's data, many of enterprise do not want to remove the information of who has been stepped aside.

Solution:

1. Implementing the data investigation on the AD objects.
2. Having more focus on Administrative objects (accounts, groups).
3. Having more focus on Service accounts and check if the services are still running.
4. Promulgating the policies for those unused objects (delete them or deactivate them).

4 Windows AD security development

This chapter aims to present some of security development in Windows AD which are concluded by author's experience. These solutions were implemented in realistic environment where the data is resided on the production database. This chapter also want to emphasize the combination between PowerShell and another open source software in this case is Zabbix. Why Zabbix? As an enterprise open source software for network and applications, and used by many large enterprises. Thence, the author is questionless about its reputation compared with other monitoring tools, with the web configuration is made to interact with users, Zabbix gives much more visually intuitive, you can typically pick up on how to configure quickly along with specific documents as such how to make a template, hostname, using macro and so on. Through web configuration, Zabbix also provides graphs and stats collection via SMTP or HTTP; thanks to this creation, Zabbix becomes an extremely awesome tool for monitoring data as well as monitoring trends.

In order to have more comprehending in Windows AD security and how Windows environment operates in general, this is the main reason makes the author go for this combination.

In another hand, these solutions are tremendously flexible and it can be implemented in every AD infrastructure. It points out which privileged groups and privileged accounts need to be manged, controlled and monitored, and also two applicable security solutions are information gathering and monitoring by using PowerShell as a main language for this development.

4.1 Privileged accounts in sensitive groups

As chapter 3 has showed privileged groups in tier 0 and also indicated the important of it in Windows AD security. Nevertheless, in Windows AD, there are few other groups that are considered sensitive groups in which the users have more permission in performing their actions in AD, and these groups need to be manged and monitored as well. Such as:

Group Names	Description
Allowed RODC Password Replication Group	Active Directory group where members can have their domain password cached on a RODC after successfully authenticating (includes user and computer accounts). The purpose of this security group is to manage a RODC password replication policy. This group has no members by default, and it results in the condition that new Read-only domain controllers do not cache user credentials.
Certificate Service DCOM Access	Active Directory group. Members of this group are allowed to connect to certification authorities in the enterprise.
Cert Publishers	Active Directory group. Members of the Cert Publishers group are authorized to publish certificates for User objects in Active Directory.
Distributed COM Users	Members of the Distributed COM Users group are allowed to launch, activate, and use Distributed COM objects on the computer. Microsoft Component Object Model (COM) is a platform-independent, distributed, object-oriented system for creating binary software components that can interact. Distributed Component Object Model (DCOM) allows applications to be distributed across locations that make the most sense to you and to the application. This group appears as a SID until the domain controller is made the primary domain controller and it holds the operations master role (also known as flexible single master operations or FSMO).
Event Log Readers	Members of this group can read event logs from local computers. The group is created when the server is

	promoted to a domain controller.
DnsAdmins	Members of DNSAdmins group have access to network DNS information. The default permissions are: Allow: Read, Write, Create All Child objects, Delete Child objects, Special Permissions.
Group Policy Creators Owners	This group is authorized to create, edit, or delete Group Policy Objects in the domain. By default, the only member of the group is Administrator.
Hyper-V Administrators	Members of the Hyper-V Administrators group have complete and unrestricted access to all the features in Hyper-V. Adding members to this group helps reduce the number of members required in the Administrators group, and further separates access.
Pre–Windows 2000 Compatible Access	Members of the Pre–Windows 2000 Compatible Access group have Read access for all users and groups in the domain.
Remote Desktop Users	The Remote Desktop Users group on an RD Session Host server is used to grant users and groups permissions to remotely connect to an RD Session Host server. This group cannot be renamed, deleted, or moved. It appears as a SID until the domain controller is made the primary domain controller and it holds the operations master role (also known as flexible single master operations or FSMO).
WinRMRemoteWMIUsers	In Windows Server 2012, the Access Denied Assistance functionality adds the Authenticated Users group to the local WinRMRemoteWMIUsers__ group. Therefore, when the Access Denied Assistance functionality is

	enabled, all authenticated users who have Read permissions to the file share can view the file share permissions. The WinRMRemoteWMIUsers_ group allows running Windows PowerShell commands remotely whereas the Remote Management Users group is generally used to allow users to manage servers by using the Server Manager console.
Protected Users	Members of the Protected Users group are afforded additional protection against the compromise of credentials during authentication processes. This security group is designed as part of a strategy to effectively protect and manage credentials within the enterprise. Members of this group automatically have non-configurable protection applied to their accounts.

Table 5. Sensitive groups [13].

These are sensitive groups which have the ability to give an account having more particular permissions to act in AD. You should consider those accounts are privileged accounts, they exist in many forms across an enterprise environment, and they pose significant security risks if not protected, managed and monitored.

These types of accounts need to be monitored frequently due to the risks it might give. Nowadays, there is many software developed for protecting these accounts either by third-party or Microsoft. It is worth to notice that you have to pay heavy price annually for the software as well as consultancy. Thereby, the section 4.2 and 4.3 will present the "cheap" solutions but it brings extremely effective in securing privileged accounts by monitoring sensitive groups in AD and it is super flexible that every AD infrastructure can straightforwardly implement it.

4.2 Privileged accounts information gathering

One of the important processes in securing Windows privileged accounts that used to be ignored in many large enterprises where they are using Active Directory, in general, is gathering information. In particular, AD environment contains a variety of information; this information makes it possible to know where devices are currently placed, their configuration, and the impact of changes to Active Directory that result from implementing defence strategy like tier 0 groups investigation.

In chapter 3, the author has listed the solution on how to apply the principles for every tier 0 groups as well as pointed out the importance and impact of those groups on Windows AD infrastructure from the security point of views. For example, members of the Domain Admins security group are authorized to administer the domain. By default, the Domain Admins group is a member of the Administrators group on all computers that have joined a domain, including the domain controllers. The Domain Admins group is the default owner of any object that is created in Active Directory for the domain by any member of the group. If members of the group create other objects, such as files, the default owner is the Administrators group [13].

Thence, understanding the inheritance reside on your infrastructure is a vital key to have a good defence strategy and solution. In order to do that, you should conduct more investigation on the data especially in this case is in Tier 0 groups and sensitive groups. The number of how many accounts are belonging to these groups along with some helpful attributes that will give you the information precisely such as Account status, Smart card required, Account expires, email and etc. Thanks to this information you can straightforwardly move to the assessment and promulgate the security policies for these accounts.

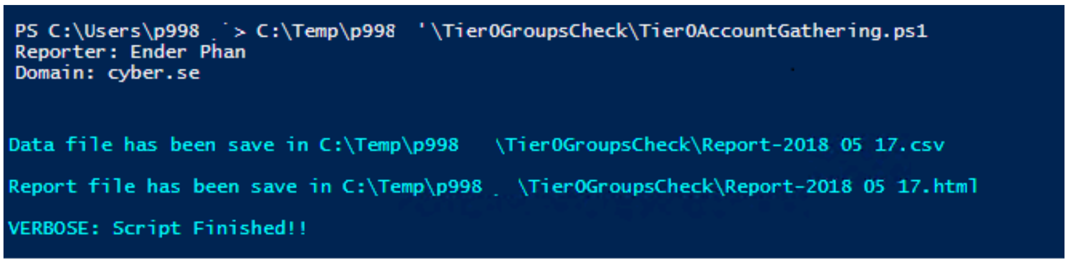
The author has developed and implemented the investigation in Swedbank. The script is written in PowerShell, Net and HTML, it queries all the accounts in the tier 0 groups and sensitive groups, along with the replicated attributes as the following:

- distinguishedName
- sAMAccountName
- mail

- lastLogonTimeStamp
- pwdLastSet
- accountExpires,
- userAccountControl
- Description

Especially, with userAccountControl attribute you can get many more accounts information like Smart Card exception, Password requires, Password never expires and etc.

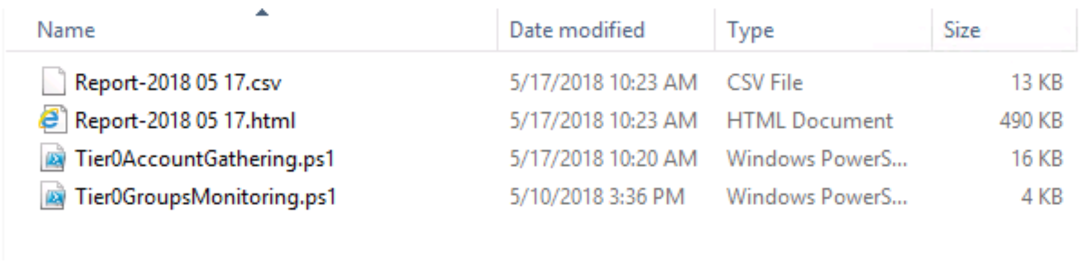
The administrators or security officers can run it manually or set it run in task schedule biweekly or bimonthly depending on the needs. Basically, if the scripts are executed manually, there are 2 parameters need to be inserted including reporter name and domain name.



```
PS C:\Users\p998 > C:\Temp\p998 '\Tier0GroupsCheck\Tier0AccountGathering.ps1
Reporter: Ender Phan
Domain: cyber.se


Data file has been save in C:\Temp\p998  \Tier0GroupsCheck\Report-2018 05 17.csv

Report file has been save in C:\Temp\p998  \Tier0GroupsCheck\Report-2018 05 17.html

VERBOSE: Script Finished!!
```

Figure 16. Tier0AccountGathering gets executed

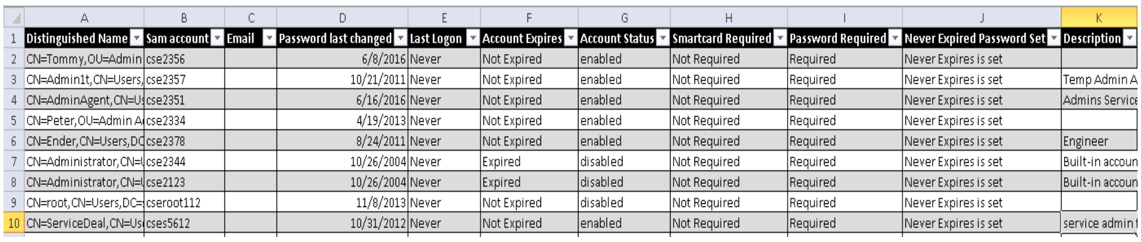
After executing successfully, it notifies that 2 files have been exported which are account information saved in csv format and also the report in html.



Name	Date modified	Type	Size
Report-2018 05 17.csv	5/17/2018 10:23 AM	CSV File	13 KB
Report-2018 05 17.html	5/17/2018 10:23 AM	HTML Document	490 KB
Tier0AccountGathering.ps1	5/17/2018 10:20 AM	Windows PowerS...	16 KB
Tier0GroupsMonitoring.ps1	5/10/2018 3:36 PM	Windows PowerS...	4 KB

Figure 17. Reports are exported

The CSV contains the account information given by built-in useful attributes, by taking advantage of SCV functionality, you can easily filter out the information you are interested in.

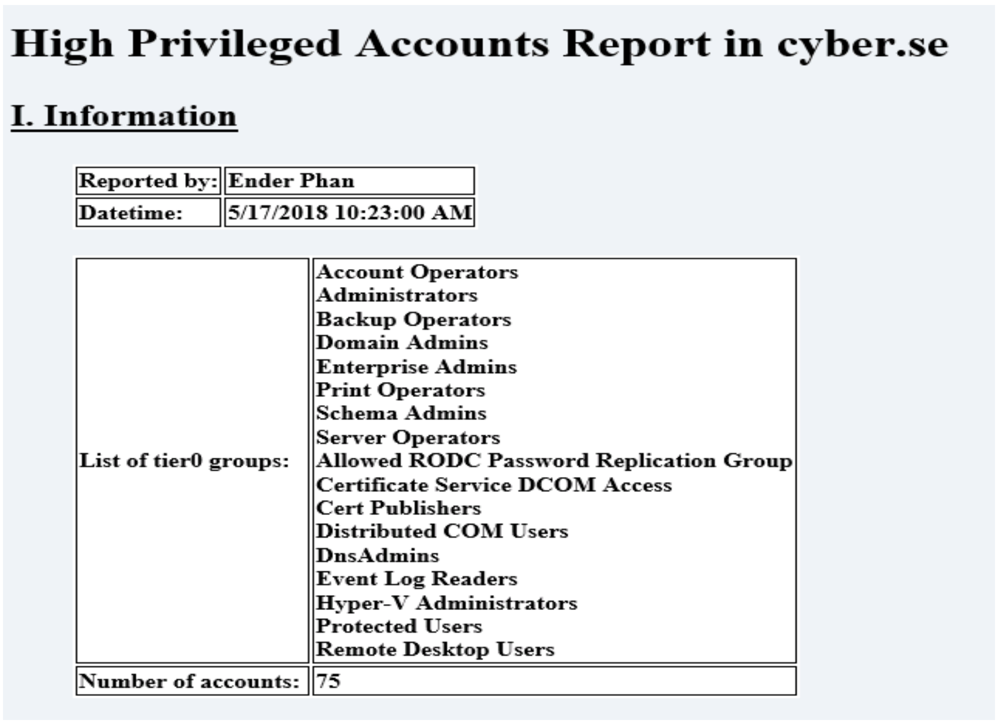


	A	B	C	D	E	F	G	H	I	J	K
1	Distinguished Name	Sam account	Email	Password last changed	Last Logon	Account Expires	Account Status	Smartcard Required	Password Required	Never Expired Password Set	Description
2	CN=Tommy,OU=Admin	cse2356		6/8/2016	Never	Not Expired	enabled	Not Required	Required	Never Expires is set	
3	CN=Admin1t,CN=Users,	cse2357		10/21/2011	Never	Not Expired	enabled	Not Required	Required	Never Expires is set	Temp Admin A
4	CN=AdminAgent,CN=U	cse2351		6/16/2016	Never	Not Expired	enabled	Not Required	Required	Never Expires is set	Admins Service
5	CN=Peter,OU=Admin A	cse2334		4/19/2013	Never	Not Expired	enabled	Not Required	Required	Never Expires is set	
6	CN=Ender,CN=Users,DC	cse2378		8/24/2011	Never	Not Expired	enabled	Not Required	Required	Never Expires is set	Engineer
7	CN=Administrator,CN=U	cse2344		10/26/2004	Never	Expired	disabled	Not Required	Required	Never Expires is set	Built-in accoun
8	CN=Administrator,CN=U	cse2123		10/26/2004	Never	Expired	disabled	Not Required	Required	Never Expires is set	Built-in accoun
9	CN=root,CN=Users,DC=	cseroot112		11/8/2013	Never	Not Expired	disabled	Not Required	Required	Never Expires is set	
10	CN=ServiceDeal,CN=Us	cses5612		10/31/2012	Never	Not Expired	enabled	Not Required	Required	Never Expires is set	service admin

Figure 18. Information of account saved in CSV format

A report in CSV format is really helpful for the investigators to filter out the fields that are interesting to them. With a detail information of the tier 0 groups account such as Sam Account, Account status, Email and description, the administrator can straightforwardly collate them with its configuration in AD as shown in figure 18.

In large enterprises, this report in CSV format is not sufficient, it just useful for technical persons. How about non-technical persons can understand and have a good assessment on the data? Thereby, the report in HTML created to assist them in viewing the data better and more general.



High Privileged Accounts Report in cyber.se

I. Information

Reported by:	Ender Phan
Datetime:	5/17/2018 10:23:00 AM

List of tier0 groups:	Account Operators Administrators Backup Operators Domain Admins Enterprise Admins Print Operators Schema Admins Server Operators Allowed RODC Password Replication Group Certificate Service DCOM Access Cert Publishers Distributed COM Users DnsAdmins Event Log Readers Hyper-V Administrators Protected Users Remote Desktop Users
Number of accounts:	75

Figure 19. Information section in HTML report

The information section in the HTML report in figure 19 shows whom has executed the script and when. It also comprises the list of tier 0 groups which have been scanned to get member those are belonged to it. The last row presents number of accounts in total. The HTML report is written in HTML and CSS;

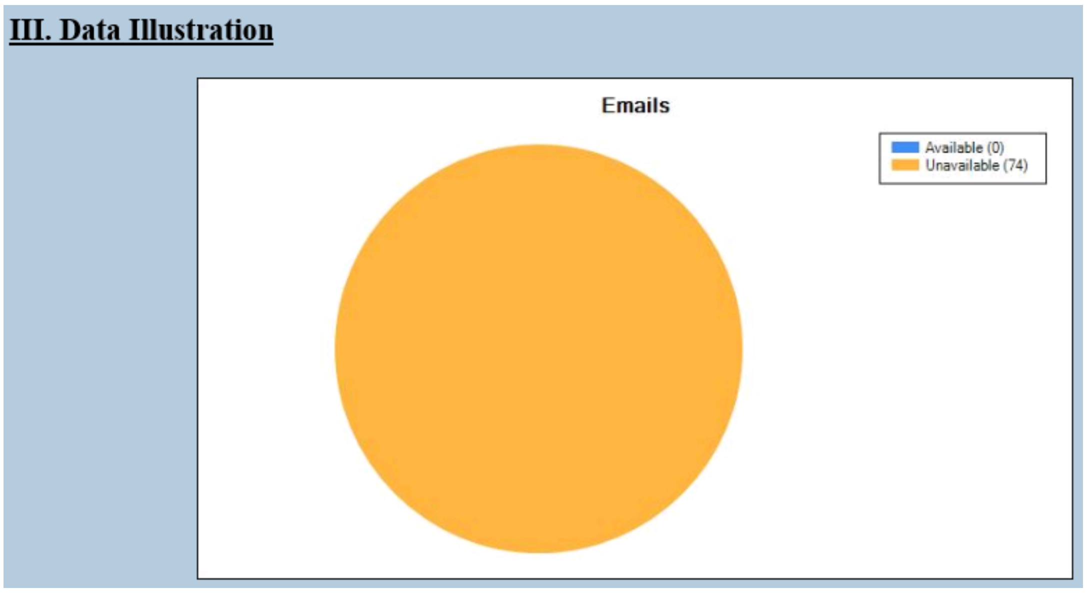


Figure 20. Illustration section in HTML report

The illustration part is created for non-technical persons or in this case is business people who is responsible for data evaluation. Based on the pie and column chart, the evaluators will be grasp the situation easier before giving out the assessments.

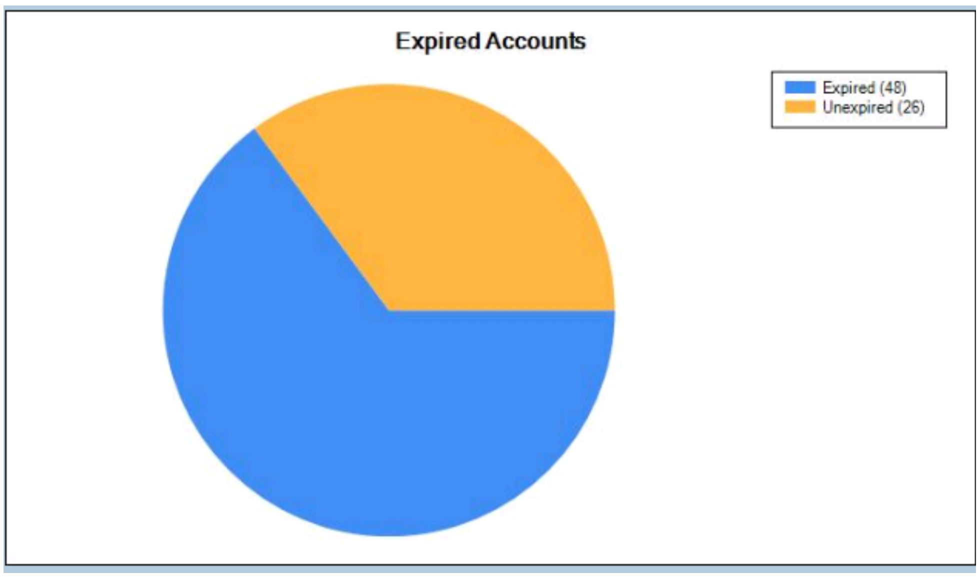


Figure 21. Expired accounts in pie chart.

The figure 21 shows us that there are 48 accounts in tier 0 groups are expired but still stay in AD meanwhile 26 accounts are unexpired. But we do not assure that those unexpired are being used or unused.

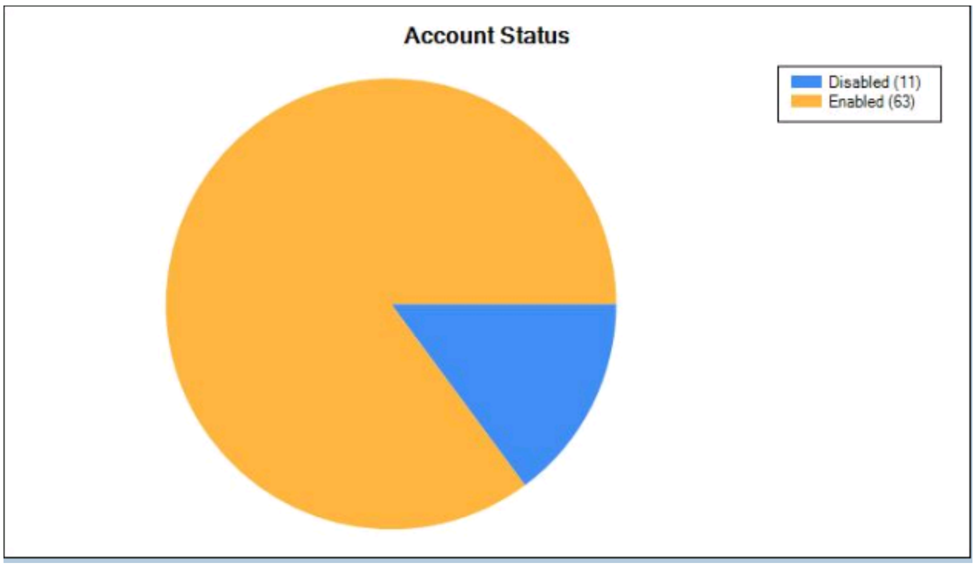


Figure 22. Account status in pie chart.

The expired account chart does not tell us that those unexpired accounts are still being used or not. Therefore, the account status attribute in figure 22 will contribute to this clarification, it describes the account are unable or disable. This status is dependent on account configuration or administrator, for example when an employee steps aside their job, the administrator will disable an account or enable it when they come back. We can easily see that there are 11 tier 0 accounts are disabled.

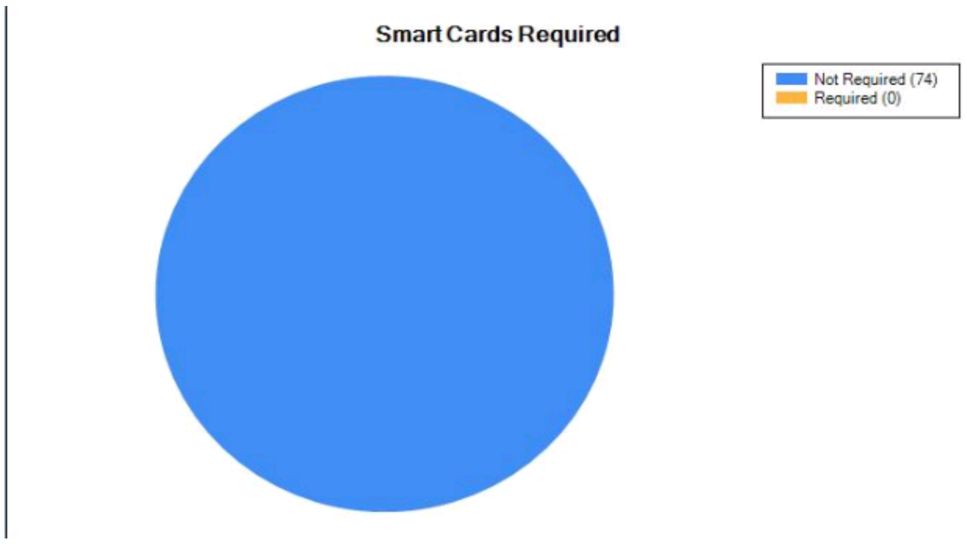


Figure 23. Smart card requires in pie chart.

The smart card exception is an important attribute for two factors authentication. Because the script executed in a test environment, thus those accounts are not required the smart card as shown in figure 23. Nevertheless, in the production environment, enforcing tier 0 accounts using the smart card for authentication is highly recommended and it must be performed immediately once you deploy AD.

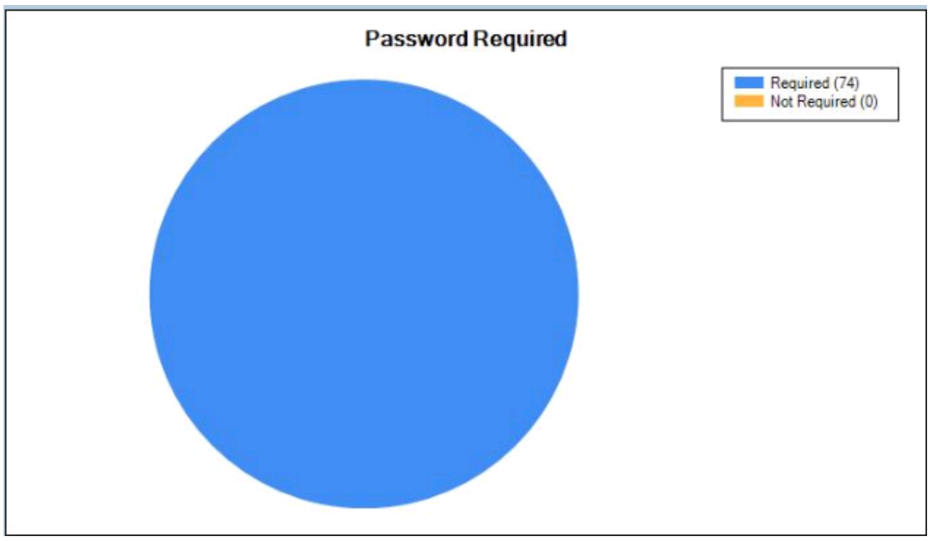


Figure 24. Password requires in pie chart.

Obviously, the figure 24 illustrates the user password requires attribute. Because smart card exception was enforced as showed in figure 25; thus, all account in tier 0 groups have a password required for authentication. This is not recommended in the production environment.

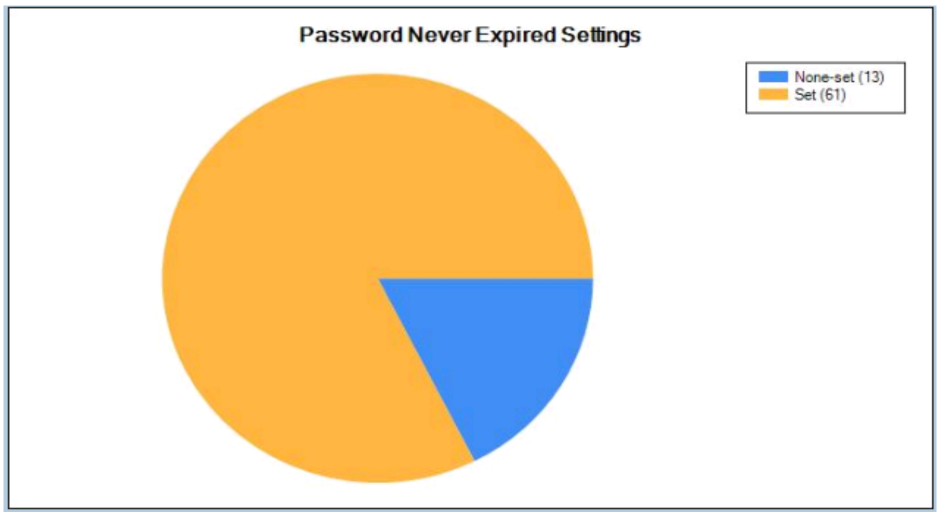


Figure 25. Password never expires in pie chart.

Password never expires is one of the most crucial attributes for AD account and privileged AD account in general. If the attacker has one of this password never expires account in AD, surely that they already have the strongest backdoor ever and unsuspicious factor that never be detected. Thus, doing the investigation on these account with this attribute is a crucial role for an administrator, she or he should be the one who knows the account is being used by an authenticated employee.

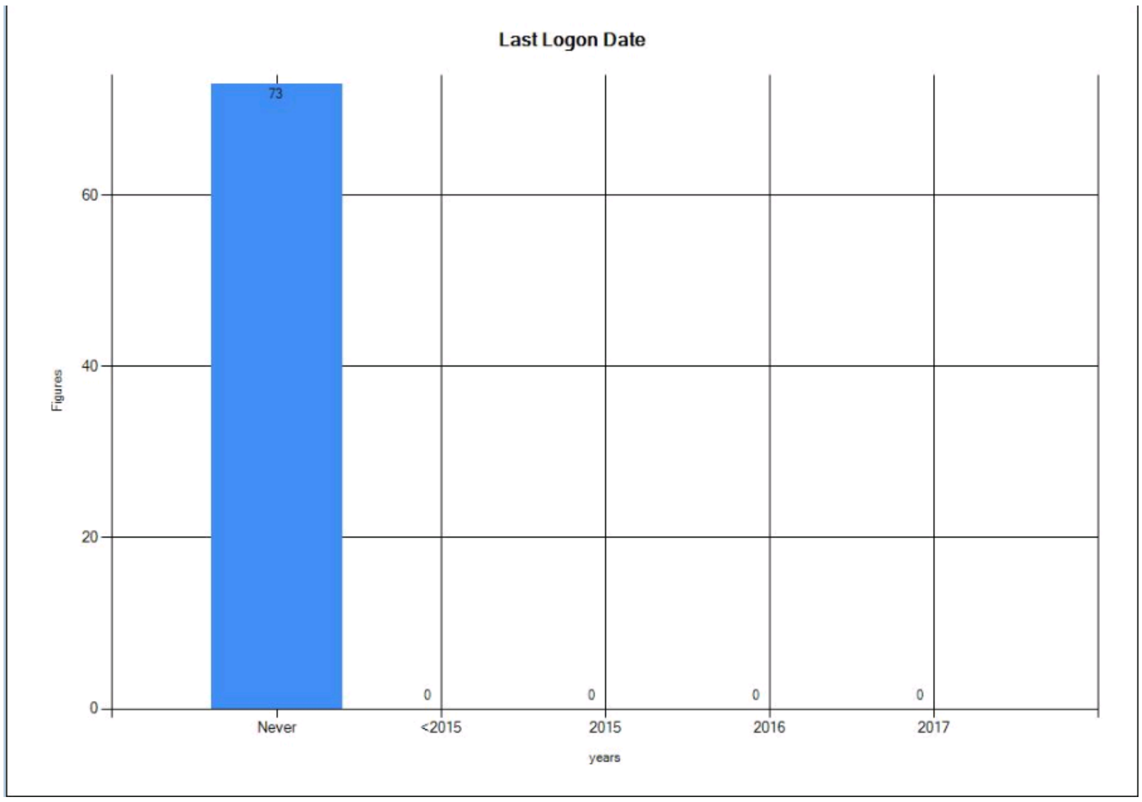


Figure 26. Last logon attribute in column chart.

Besides the pie chart, column chart is also used to illustrate the data, depending on the information of the account. The figure 26 shows us really informative chart, those sensitive accounts have never logged on. This information should not happen in production server, because in the test environment, this attribute is not set for the account; however, this attribute presents helpful message regard the last logon date of the account, thanks to this, we can easily know which accounts have the most recent logon.

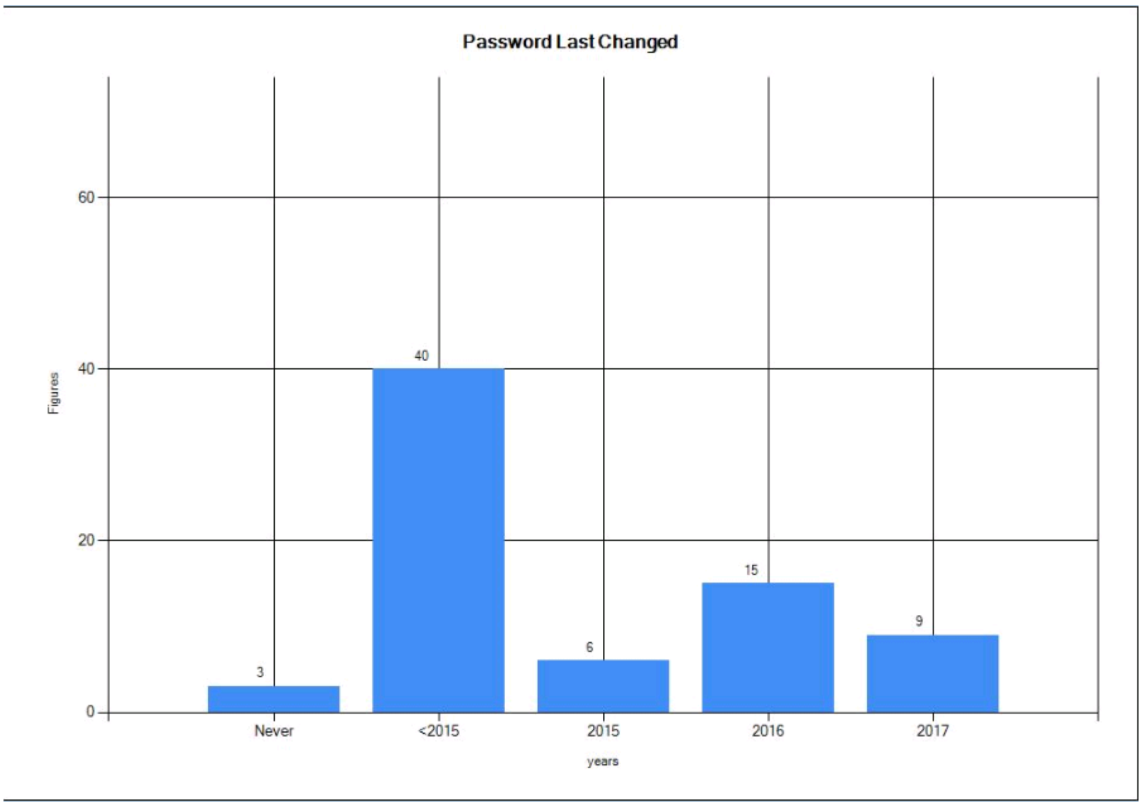


Figure 27. Password last changed in column chart.

In order to serve for the investigation, this attribute tells when the last time the account had changed their password. As shown in figure 27, there are 3 accounts had never changed its password. If all tier 0 accounts are not enforced smart card for authentication, this will not be a normal case. The privileged account's password should be changed at least once in a month to preventing attackers using it as a back connection. Thus, this chart shows us that more than 88% account in tier 0 groups have not changed their password before 2017, this would not be a good sign if it was shown in production server.

4.3 Monitoring development using PowerShell and Zabbix

First of all, there are many software in the wild or commercial software developed by Microsoft for Active Directory monitoring such as ATA, RSA and so on. We cannot argue the benefits it gives as well as the quality of paid software, indeed. Nevertheless, in this particular case, the author wishes to emphasize of how to make the advantage of using available resources without paying heavy money but also bringing significantly effective result.

Two main resources here are PowerShell and open source monitoring software Zabbix. This is great combination by using PowerShell as a tool to interact with AD assets then manipulating Zabbix to propagate that information to the administrator. Given the possibility to modify the needs; thus, this combination has a big advantage for you to change and make it more suitable for your AD infrastructure in terms of monitoring privileged accounts.

Setting up testing environment:

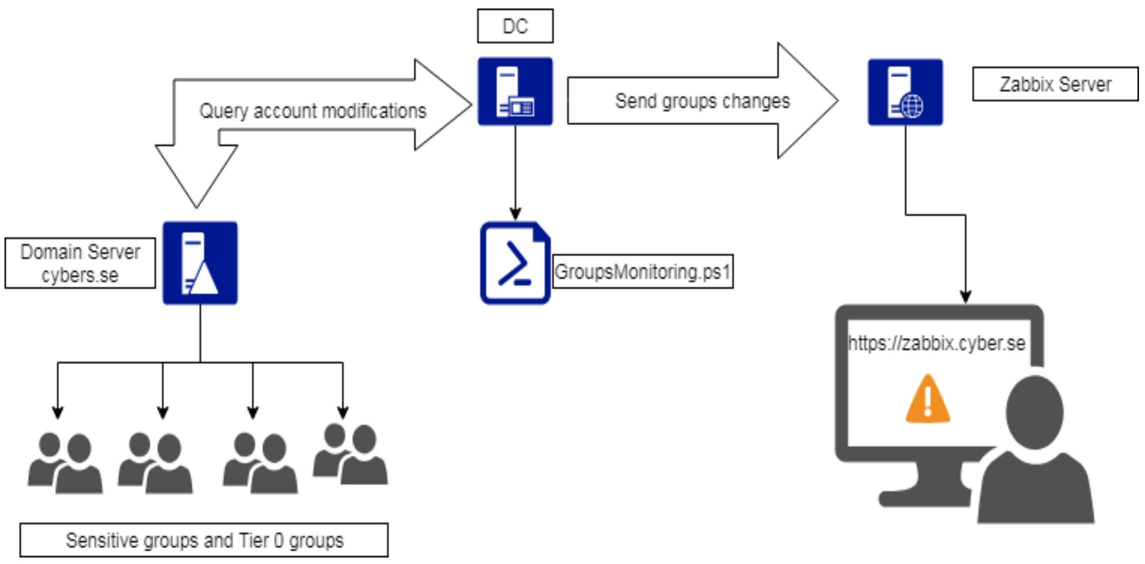
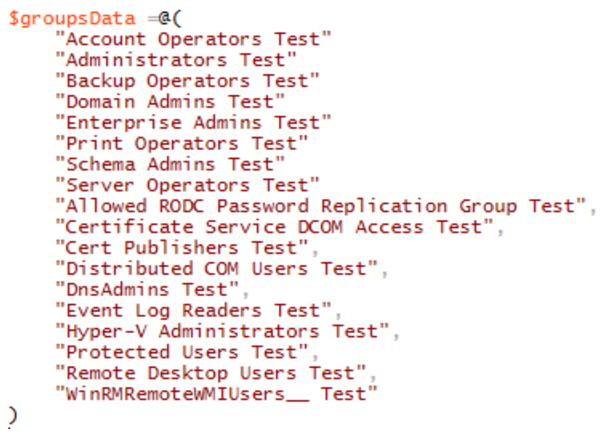


Figure 28. Privileged Accounts Monitoring Process.

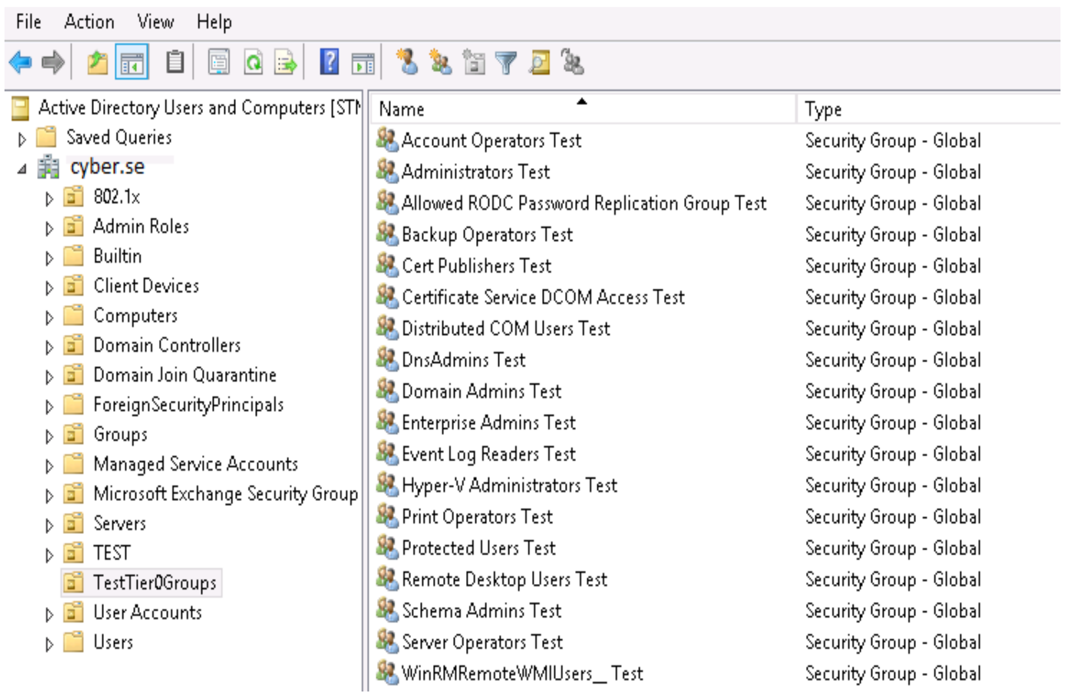
The author set up a testing environment which includes an AD domain server with cyber.se as a domain name. Cyber.se contains sensitive groups and tier 0 groups with default elevated rights as showing in this picture below:



```
$groupsData =@(
    "Account Operators Test"
    "Administrators Test"
    "Backup Operators Test"
    "Domain Admins Test"
    "Enterprise Admins Test"
    "Print Operators Test"
    "Schema Admins Test"
    "Server Operators Test"
    "Allowed RODC Password Replication Group Test",
    "Certificate Service DCOM Access Test",
    "Cert Publishers Test",
    "Distributed COM Users Test",
    "DnsAdmins Test",
    "Event Log Readers Test",
    "Hyper-V Administrators Test",
    "Protected Users Test",
    "Remote Desktop Users Test",
    "WinRMRemoteWMIUsers__ Test"
)
```

Figure 29. Sensitive groups and tier 0 groups.

Those sensitive groups and tier 0 groups which reside on AD with built-in groups. In this case, for testing purpose, the author created simulated sensitive groups and tier 0 groups with “test” after each group as the following naming convention.



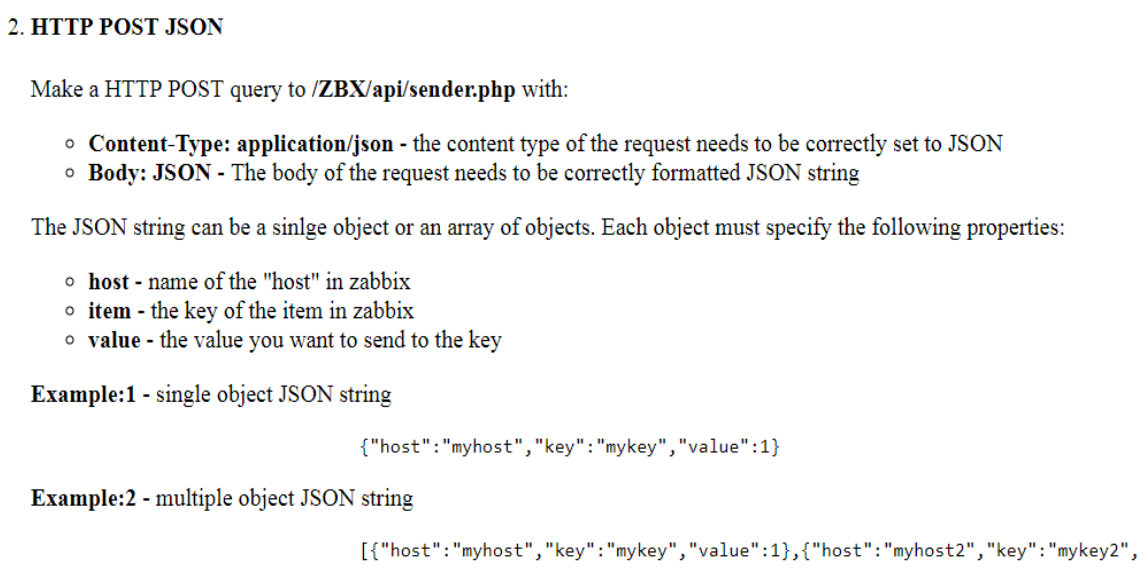
Active Directory Users and Computers [STN
- Saved Queries
- cyber.se
  - 802.1x
  - Admin Roles
  - Builtin
  - Client Devices
  - Computers
  - Domain Controllers
  - Domain Join Quarantine
  - ForeignSecurityPrincipals
  - Groups
  - Managed Service Accounts
  - Microsoft Exchange Security Group
  - Servers
  - TEST
  - TestTier0Groups
  - User Accounts
  - Users

Name	Type
Account Operators Test	Security Group - Global
Administrators Test	Security Group - Global
Allowed RODC Password Replication Group Test	Security Group - Global
Backup Operators Test	Security Group - Global
Cert Publishers Test	Security Group - Global
Certificate Service DCOM Access Test	Security Group - Global
Distributed COM Users Test	Security Group - Global
DnsAdmins Test	Security Group - Global
Domain Admins Test	Security Group - Global
Enterprise Admins Test	Security Group - Global
Event Log Readers Test	Security Group - Global
Hyper-V Administrators Test	Security Group - Global
Print Operators Test	Security Group - Global
Protected Users Test	Security Group - Global
Remote Desktop Users Test	Security Group - Global
Schema Admins Test	Security Group - Global
Server Operators Test	Security Group - Global
WinRMRemoteWMIUsers__Test	Security Group - Global

Figure 30. Set up groups in Cyber.se

The figure 30 describes those sensitive groups and tier 0 groups had been created in TestTier0Group Organization Unit.

Zabbix installation:



2. **HTTP POST JSON**

Make a HTTP POST query to **/ZBX/api/sender.php** with:

- **Content-Type: application/json -** the content type of the request needs to be correctly set to JSON
- **Body: JSON -** The body of the request needs to be correctly formatted JSON string

The JSON string can be a sinlge object or an array of objects. Each object must specify the following properties:

- **host -** name of the "host" in zabbix
- **item -** the key of the item in zabbix
- **value -** the value you want to send to the key

Example:1 - single object JSON string

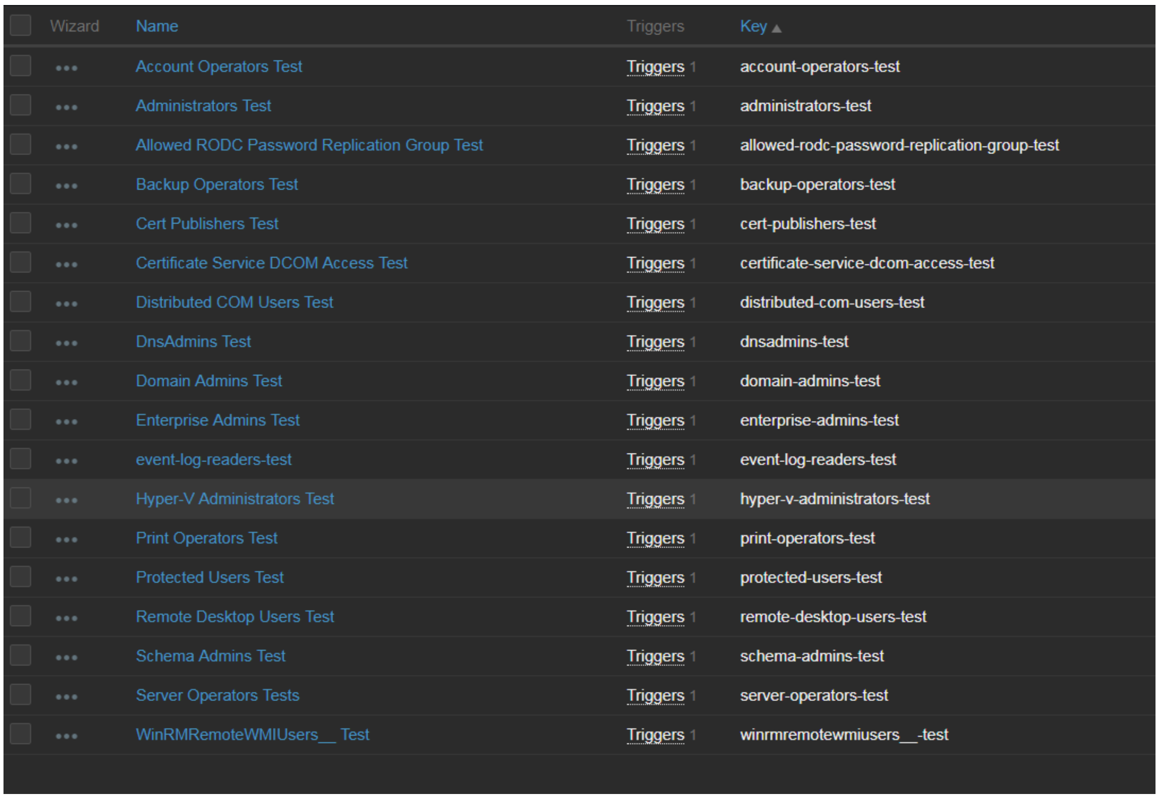
```
{"host":"myhost","key":"mykey","value":1}
```

Example:2 - multiple object JSON string

```
[{"host":"myhost","key":"mykey","value":1},{"host":"myhost2","key":"mykey2",
```

Figure 31. Zabbix API

After installing Zabbix application, the Zabbix API is the crucial factor to interact with PowerShell script and propagate the information to Zabbix Server. As shown in figure 31, the content of data sent by PowerShell script needs to be correctly set to JSON. The JSON string can be a single object or an array of objects. Each object must specify the host, item, and value which can be configured through Zabbix Web Application.

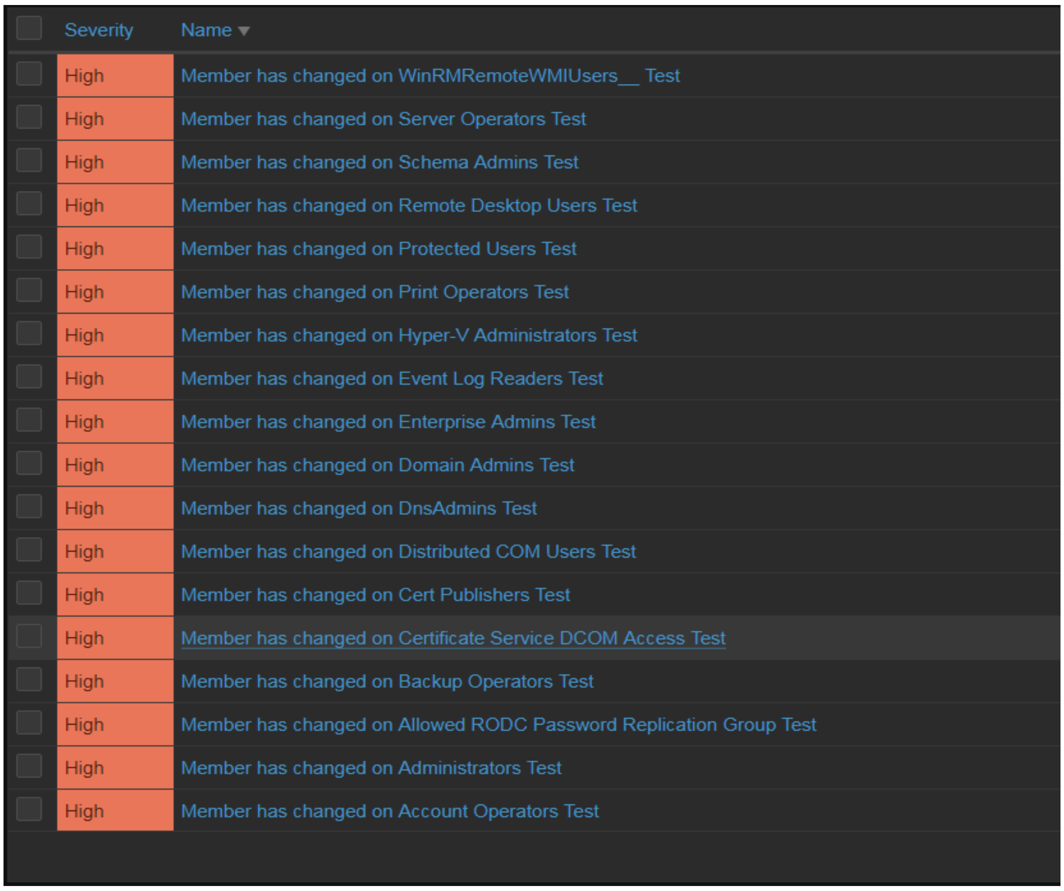


Wizard	Name	Triggers	Key ▲
•••	Account Operators Test	Triggers 1	account-operators-test
•••	Administrators Test	Triggers 1	administrators-test
•••	Allowed RODC Password Replication Group Test	Triggers 1	allowed-rodc-password-replication-group-test
•••	Backup Operators Test	Triggers 1	backup-operators-test
•••	Cert Publishers Test	Triggers 1	cert-publishers-test
•••	Certificate Service DCOM Access Test	Triggers 1	certificate-service-dcom-access-test
•••	Distributed COM Users Test	Triggers 1	distributed-com-users-test
•••	DnsAdmins Test	Triggers 1	dnsadmins-test
•••	Domain Admins Test	Triggers 1	domain-admins-test
•••	Enterprise Admins Test	Triggers 1	enterprise-admins-test
•••	event-log-readers-test	Triggers 1	event-log-readers-test
•••	Hyper-V Administrators Test	Triggers 1	hyper-v-administrators-test
•••	Print Operators Test	Triggers 1	print-operators-test
•••	Protected Users Test	Triggers 1	protected-users-test
•••	Remote Desktop Users Test	Triggers 1	remote-desktop-users-test
•••	Schema Admins Test	Triggers 1	schema-admins-test
•••	Server Operators Tests	Triggers 1	server-operators-test
•••	WinRMRemoteWMIUsers__Test	Triggers 1	winrmremotewmiusers__-test

Figure 32. Configure Host name, Item name for each group in Zabbix.

In order to identify the changes of members in each group, hostname created to contain all of the items which present for each group in Zabbix. Every item name has its own key value, these key values are also specified in PowerShell Script in order map the corresponding groups' data to the right place.

The hostname should be following naming convention in order to not give the confusion when you have more than 50 hostnames with different things to monitor. For example, the hostname, in this case: InfraDir-ADCore-Test-Cyber. With this hostname, you can easily see the purpose of the monitoring they are performing for. Applying this to key values as well, it should be describing its item names in computer reading format as shown in figure 32.



☐	Severity	Name ▾
☐	High	Member has changed on WinRMRemoteWMIUsers__ Test
☐	High	Member has changed on Server Operators Test
☐	High	Member has changed on Schema Admins Test
☐	High	Member has changed on Remote Desktop Users Test
☐	High	Member has changed on Protected Users Test
☐	High	Member has changed on Print Operators Test
☐	High	Member has changed on Hyper-V Administrators Test
☐	High	Member has changed on Event Log Readers Test
☐	High	Member has changed on Enterprise Admins Test
☐	High	Member has changed on Domain Admins Test
☐	High	Member has changed on DnsAdmins Test
☐	High	Member has changed on Distributed COM Users Test
☐	High	Member has changed on Cert Publishers Test
☐	High	Member has changed on Certificate Service DCOM Access Test
☐	High	Member has changed on Backup Operators Test
☐	High	Member has changed on Allowed RODC Password Replication Group Test
☐	High	Member has changed on Administrators Test
☐	High	Member has changed on Account Operators Test

Figure 33. Configure triggers for each group.

Trigger is an essential part of this monitoring solution. You should consider these groups are able to grand elevated permissions for the accounts in AD thereby setting up severity for the changing should be also considered high or even critical. The name of these triggers will tell you what is happening if groups are being modified; Thus, you should name it as meaningful as possible, those will impact on how quick you react on the changes.

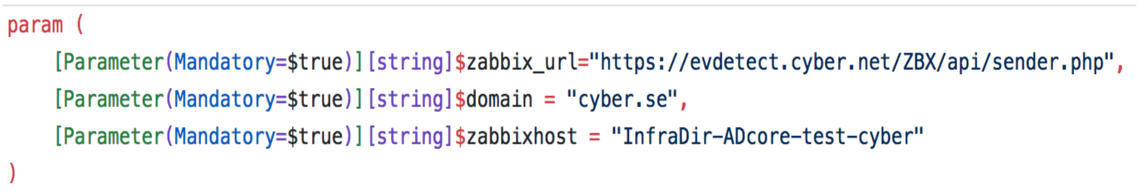
PowerShell development:

The Get-ADGroup cmdlet gets a group or performs a search to retrieve multiple groups from an Active Directory. This cmdlet gets a default set of group object properties. To get additional properties use the Properties parameter [21].

```
function getMember($groupName){
    $member = @(get-ADGroup -Identity $groupName -Properties * -server "cyber.se"| select -ExpandProperty member)
    return ($member|sort) -join ", "
}
```

Figure 34. Get member function returns Get-ADGroup member

As shown in figure 34, the Get-ADGroup cmdlet is performed selecting member of groups. Variable $groupName is an array list which contain sensitive groups and tier 0 groups as described in figure 29.

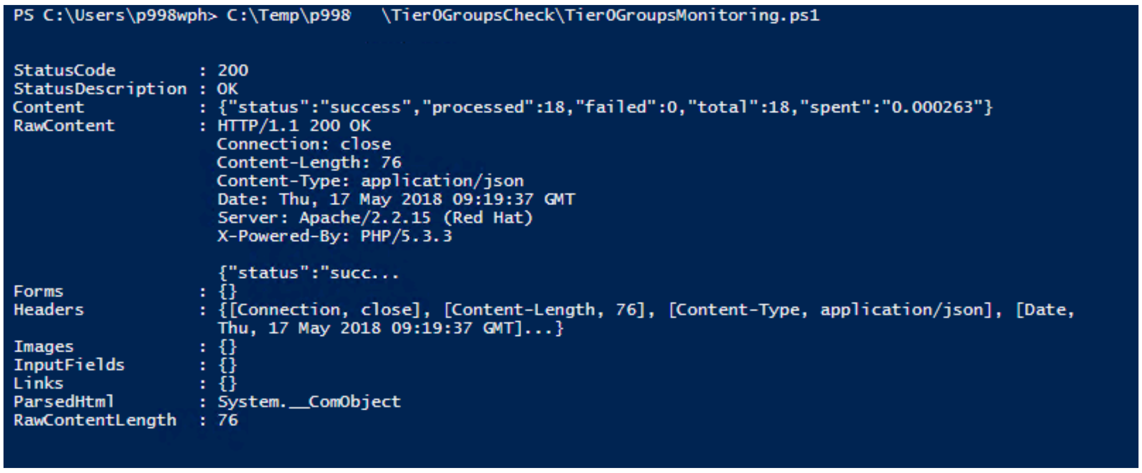


```
param (
    [Parameter(Mandatory=$true)][string]$zabbix_url="https://evdetect.cyber.net/ZBX/api/sender.php",
    [Parameter(Mandatory=$true)][string]$domain = "cyber.se",
    [Parameter(Mandatory=$true)][string]$zabbixhost = "InfraDir-ADcore-test-cyber"
)
```

Figure 35. Scripts parameters

Explained in Zabbix configuration part, Zabbix API is installed to help to propagate the information from groups in AD to Zabbix, and Zabbix API presents as a web application. In addition, Zabbix host also needs to specify as a mandatory parameter which identify the host where the data should arrive before going to every item names.

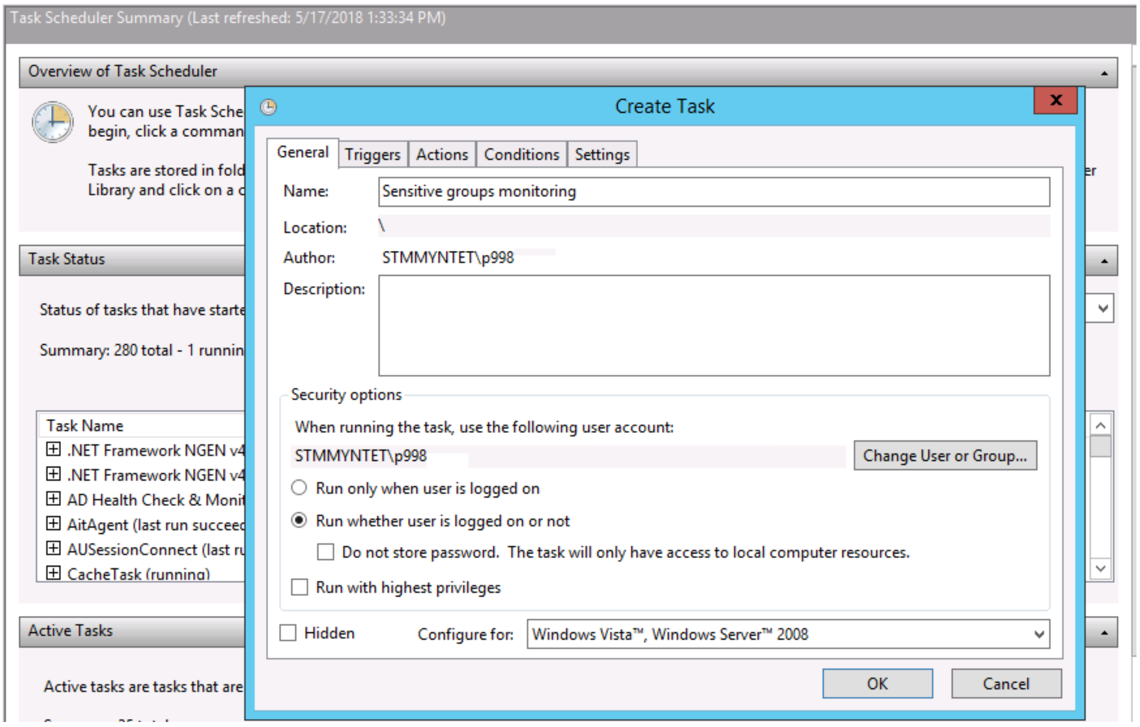
Implementation:



```
PS C:\Users\p998wph> C:\Temp\p998   \Tier0GroupsCheck\Tier0GroupsMonitoring.ps1


StatusCode        : 200
StatusDescription : OK
Content           : {"status":"success","processed":18,"failed":0,"total":18,"spent":"0.000263"}
RawContent        : HTTP/1.1 200 OK
                    Connection: close
                    Content-Length: 76
                    Content-Type: application/json
                    Date: Thu, 17 May 2018 09:19:37 GMT
                    Server: Apache/2.2.15 (Red Hat)
                    X-Powered-By: PHP/5.3.3

                    {"status":"succ...
Forms             : {}
Headers           : {[Connection, close], [Content-Length, 76], [Content-Type, application/json], [Date,
                    Thu, 17 May 2018 09:19:37 GMT]...}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : System.__ComObject
RawContentLength  : 76
```

Figure 36. Execute GroupsMonitoring.ps1

There are many ways to execute the script, as shown in figure 36, the script is executed manually from PowerShell console. The status presents 200 which means the request has succeeded. The content shows that there are 18 processes have been sent with status is success, these 18 processes represent for 18 groups with its item names specified in the code.



Task Scheduler Summary (Last refreshed: 5/17/2018 1:33:34 PM)

Create Task

General | Triggers | Actions | Conditions | Settings

Name: Sensitive groups monitoring

Location: \

Author: STMMYNTET\p998

Description:

Security options

When running the task, use the following user account:

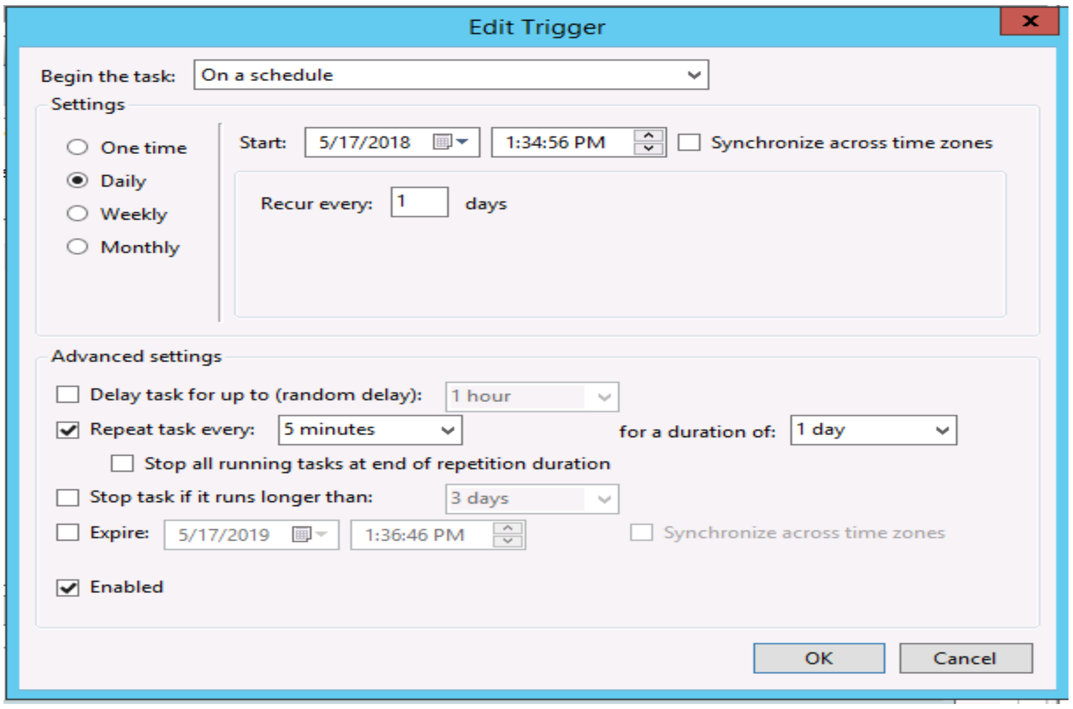
STMMYNTET\p998 [Change User or Group...]

- ○ Run only when user is logged on
- ● Run whether user is logged on or not
- ☐ Do not store password. The task will only have access to local computer resources.
- ☐ Run with highest privileges

☐ Hidden Configure for: Windows Vista™, Windows Server™ 2008

Figure 37. Create task in Task Scheduler.

For monitoring purpose, running the script manually is not proper method to perform due to the effectiveness and convenience of it, described in figure 37, the author advises to take the advantage of using Task Scheduler, this is one of built-in program in Windows which helps us to schedule our task and execute it under configuration.



Edit Trigger

Begin the task: On a schedule

Settings

- ○ One time
- ● Daily
- ○ Weekly
- ○ Monthly

Start: 5/17/2018 1:34:56 PM ☐ Synchronize across time zones

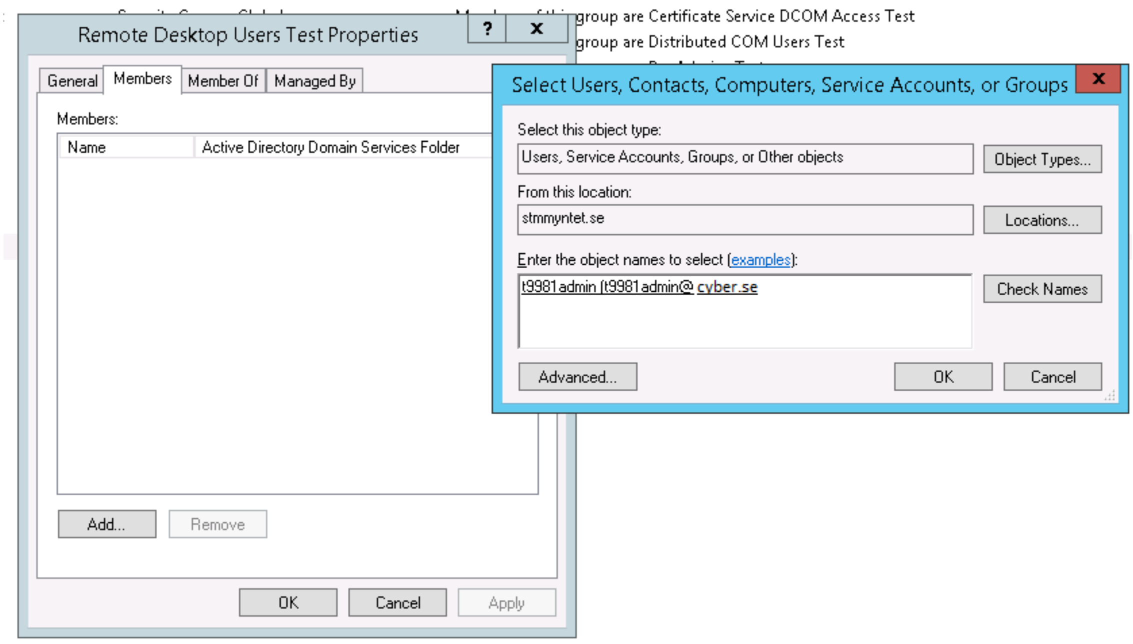
Recur every: 1 days

Advanced settings

- ☐ Delay task for up to (random delay): 1 hour
- ☑ Repeat task every: 5 minutes for a duration of: 1 day
  - ☐ Stop all running tasks at end of repetition duration
- ☐ Stop task if it runs longer than: 3 days
- ☐ Expire: 5/17/2019 1:36:46 PM ☐ Synchronize across time zones
- ☑ Enabled

Figure 38. Edit trigger for GroupsMonitoring.ps1

The trigger is important part needs to be mentioned, this decides when the GroupsMonitoring.ps1 get executed after the last execution. In this case, the author has set daily trigger and the period of the last execution to the next execution is 5 minutes, which means after 5 minutes the sensitive groups and tier 0 groups will be scanned and sent the information out to Zabbix as shown in figure 38.


group are Certificate Service DCOM Access Test
group are Distributed COM Users Test

Remote Desktop Users Test Properties

General | Members | Member Of | Managed By

Members:

Name	Active Directory Domain Services Folder

Select Users, Contacts, Computers, Service Accounts, or Groups

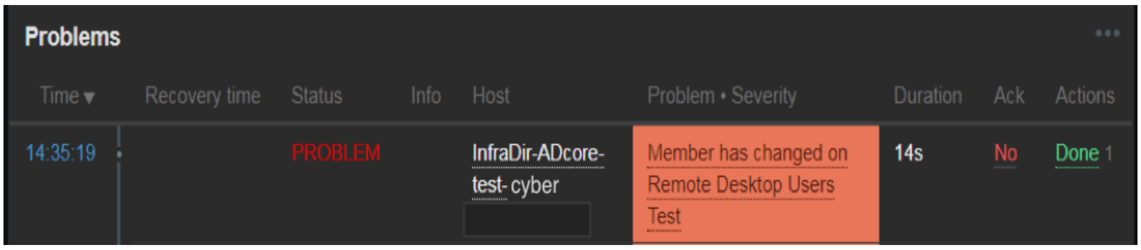
Select this object type: Users, Service Accounts, Groups, or Other objects

From this location: stmmyntet.se

Enter the object names to select (examples): t9981admin (t9981admin@ cyber.se

Figure 39. Add member to monitored group.

Figure 39 shows the scenario that the account t9981admin is being added to Remote Desktop Users Test group by an attacker.


Problems

Time ▾	Recovery time	Status	Info	Host	Problem • Severity	Duration	Ack	Actions
14:35:19		PROBLEM		InfraDir-ADcore-test- cyber	Member has changed on Remote Desktop Users Test	14s	No	Done 1

Figure 40. Zabbix alerts the change.

After the attacker added the new account to the Remote Desktop Users Test group, meanwhile this group has ability to elevate the permission for an account. From Zabbix monitoring website, the administrator can easily observe the change alerted in the problems section in figure 40. The alert shows very specifically what has been changed and where the change happened.

Given out the groups in AD which are able to grand elevated rights for the accounts that need to be controlled, managed and monitored. In order to demonstrate the importance of the necessary, the author would like to emphasize the availability of taking the advantages by using the power of PowerShell to assist us in this purpose. Information gathering brings us tremendous benefits to having a good control over the critical assets which are always being a primary target of attackers, knowing your properties well is a best way to have a proper assessment for security strategy. Moreover, giving monitoring solution helps organizations not to pay a heavy money for an obtainable development. Combining PowerShell and Zabbix could be a strong weapon to identify the abnormal behavior on-going with privileged accounts.

5 Best practice

Organizations can manage and secure their privileged accounts in Windows AD environment through use of the best practices listed in this chapter. The best practices are included by the author’s experience and also valuable recommendations from privileged access security company CyberArk. Many of these practices require only process changes, while others may involve tools or solution to implement. To help you determine which actions give you the most benefit [2].

Inventory and reduce the number of privileged accounts: knowing how many accounts are present in the environment and where they are being a critical first step in making informed risk decisions and protecting the accounts. Once inventoried, privileged accounts should be reviewed and unnecessary accounts should be deleted to reduce the overall number of accounts requiring management [2].

Secondary accounts: an employee should have 2 separate accounts to perform his/her tasks. It helps to reduce the risk it might give to the organization when the attacker can abuse his/her account to login into tier 0 or tier 1 when the user accidently executes malicious file from browsing or reading e-mail for an instance.

Enforcing least privileged for standard user accounts: the identity team should have a policy for standard account when they request for any permissions as well as the distinct processes for them to get approval. This is significant step a company can take towards enhancing the security of their AD assets.

Store password securely: It is imperative organization store their privileged passwords in the most secure, encrypted vaulting system available. The use of envelops, spreadsheets, flat files, etc. for the storage of privileged account information should be eliminated [2].

Create a process for on- and off-boarding employees that have privileged account access: employees should understand the responsibility that comes with privileged access and be trained in existing corporate policies before administrative access is granted. Access should routinely be reviewed to ensure privileged access is still required [2]. Those privileged accounts no longer in used should be disabled and changed the password immediately.

Eliminate the practice of accounts that have non-expiring passwords: passwords should be changed on a regular schedule to reduce their vulnerability to password cracking tools and password sharing between employees [2].

Password complexity and password age policy: not only privileged accounts should deploy these two policies but also regular accounts. The complexity of password should be strong enough for an attacker not easy to crack and predict, and it should not be so complex in that it encourages bad behavior such as writing down password [2]. Privileged password should be changed on a regular schedule as well, between 30 to 60 days is an appropriate period for the change.

Implement automated password verification and reconciliation to ensure that the passwords of record are current on all systems: automation is critical when manging privileged identities; new accounts are constantly created and deleted, requiring an automated system to manage and verify passwords [2].

Privileged accounts information gathering: as shown in chapter 4, the author emphasizes the importance of implementing privileged accounts information gathering. This should be performed on a regular schedule in order to knowing your evaluated users in current state then on providing solutions on those accounts that are no longer active.

Proactively detect malicious behavior: as shown in chapter 4, monitoring member of sensitive groups and tier 0 groups help to detect and alert on anomalous privileged user behavior, this is a critical layer in a best-in-class privileged accounts security strategy [2].

6 Summary

The final chapter of this thesis presents a summary of the research carried out in this work. Section 6.1 provides general conclusions, section 6.2 presents discussions arising from this work and areas of future work.

6.1 General conclusions

We have to admit that the attackers exploit valid credentials or in another word is privileged account almost 100% of the time. It points out the importance of protecting this kind of assets in your network as well as in Windows AD generally. Many large companies in every sector of the economy is susceptible to the risk of having their own privileged accounts exploited. Therefore, in this thesis, effectiveness of manipulating security principles, security development from a number of sources and derived from practices designed in order to protect Windows privileged accounts against compromise as a case study.

From the achievement, the author has aimed to point out not only the effectiveness of applying those Windows AD principle but also show you how to take the advantage of using free sources to develop a self-built weapon. Additionally, the author has also contributed one more AD principle besides others recommended by Microsoft, privileged accounts cleaning-up should be represented as a regular schedule task for an administrator understand their inheritances better thereby reducing the risks of inactive accounts. More importantly, in the practical part of the thesis, the author has built up a Windows security simulation environment, where the author had opportunity to weaponize PowerShell to perform privileged accounts information gathering and combine it with Zabbix open source software in monitoring. From this implementation, the best practices have given out, as an organization look to manipulate a solution to proactively protect and monitor privileged accounts.

6.2 Future works

Despite the effectiveness and benefits of protecting privileged accounts methods in the thesis, there is also some issues have been identified for possible future work.

Service accounts investigation: Service accounts is one of privileged account types in Windows, it can be privileged local or domain accounts that are used by an application or service to interact with the operating system [2]. Due to the complexity of changing the password of service accounts when the local service accounts can interact with a variety of Windows components. This challenge often leads to a common practice of rarely changing service account passwords which represents a significant risk across an enterprise. Many attack vectors used to abuse the service accounts to exploit such as Kerberos Golden Ticket, Kerbroasting and so on. Giving the solutions for protecting service accounts and privilege accounts in general is imperative to contribute security strategy in Windows environment.

References

[1] J. Reis, “learnthat.com,” [Online]. Available: http://learnthat.com/introduction-to-active-directory/.

[2] CyberArk, “lp.cyberark.com,” [Online]. Available: https://lp.cyberark.com/rs/cyberarksoftware/images/wp-securing-privileged-accounts-best-practice-guide-04-15-2014-en.pdf.

[3] Microsoft, “docs.microsoft.com,” 31 05 2017. [Online]. Available: https://docs.microsoft.com/en-us/Windows-server/identity/ad-ds/plan/security-best-practices/reducing-the-active-directory-attack-surface.

[4] C. Tozzi, “www.channelfutures.com,” 25 July 2016. [Online]. Available: http://www.channelfutures.com/open-source/reasons-organizations-opt-not-use-open-source-software.

[5] Microsoft, “cloudblogs.microsoft.com,” 22 08 2017. [Online]. Available: https://cloudblogs.microsoft.com/Windowsserver/2017/08/22/now-available-Windows-server-2016-security-guide/.

[6] R. Smith, “www.petri.com,” 24 October 2017. [Online]. Available: https://www.petri.com/use-microsofts-active-directory-tier-administrative-model.

[7] Microsoft, “docs.microsoft.com,” 10 12 2016. [Online]. Available: https://docs.microsoft.com/en-us/Windows-server/identity/securing-privileged-access/securing-privileged-access-reference-material.

[8] arubanetworks, “www.arubanetworks.com,” [Online]. Available: http://www.arubanetworks.com/assets/so/SO_UEBA_UseCase_CompromisedUsers.pdf.

[9] Kaspersky, “www.kaspersky.com,” unknow unknow unknow. [Online]. Available: https://www.kaspersky.com/blog/incident-response-report/.

[10] Microsoft, “docs.microsoft.com,” 05 06 2018. [Online]. Available: https://docs.microsoft.com/en-us/advanced-threat-analytics/suspicious-activity-guide.

[11] Microsoft, “cloudblogs.microsoft.com,” 19 January 2017. [Online]. Available: https://cloudblogs.microsoft.com/enterprisemobility/2017/01/19/introducing-microsoft-advanced-threat-analytics-for-your-datacenter/.

[12] Micosoft, “cloudblogs.microsoft.com,” 09 January 2017. [Online]. Available: https://cloudblogs.microsoft.com/enterprisemobility/2017/01/09/eliminating-plaintext-passwords-with-microsoft-advanced-threat-analytics-using-ldap/.

[13] S. Metcalf, “adsecurity.org,” 14 june 2017. [Online]. Available: http://adsecurity.org/?p=3658.

[14] Microsoft, “technet.microsoft.com,” February 2007. [Online]. Available: https://technet.microsoft.com/en-us/library/2007.02.activedirectory.aspx.

[15] Defcon.ru, “defcon.ru,” 08 06 2017. [Online]. Available: https://defcon.ru/network-security/4672/.

[16] Micosoft, AD Delegation, Tallinn: Micosoft, 2017.

[17] E. Zhang, “digitalguardian.com,” 23 october 2017. [Online]. Available: https://digitalguardian.com/blog/what-role-based-access-control-rbac-examples-benefits-and-more.

[18] Micosoft, “docs.microsoft.com,” 31 05 2017. [Online]. Available: https://docs.microsoft.com/en-us/Windows-server/identity/ad-ds/plan/security-best-practices/appendix-b--privileged-accounts-and-groups-in-active-directory.

[19] Microsoft, “docs.microsoft.com,” 09 06 2016. [Online]. Available: https://docs.microsoft.com/en-us/Windows-server/identity/securing-privileged-access/privileged-access-workstations.

[20] Netwrix, “www.netwrix.com,” [Online]. Available: https://www.netwrix.com/active_directory_group_management.html.

[21] Microsoft, “docs.microsoft.com,” [Online]. Available: https://docs.microsoft.com/en-us/powershell/module/addsadministration/get-adgroup?view=win10-ps.

[22] L. S. Sterling, The Art of Agent-Oriented Modeling, London: The MIT Press, 2009.

[23] J. Policelli, “www.networkworld.com,” 21 05 2009. [Online]. Available: https://www.networkworld.com/article/2255953/infrastructure-management/chatper-1--introduction-to-active-directory-domain-services.html.

[24] tanium.com, “tanium.com,” [Online]. Available: https://tanium.com/blog/whats-old-new-detecting-office-macro-malware-tanium/index.html.

[25] BeyondTrust, “Challenges of Managing Privileged Access on Windows and Servers,” Darren Mar-Elia, 2013.

[26] Microsoft, “docs.microsoft.com,” 31 05 2017. [Online]. Available: https://docs.microsoft.com/en-us/Windows-server/identity/ad-ds/plan/security-best-practices/implementing-least-privilege-administrative-models.

[27] Microsoft, 31 05 2017. [Online]. Available: https://docs.microsoft.com/en-us/Windows-server/identity/ad-ds/manage/component-updates/introduction.

[28] G. Olsen, 08 2006. [Online]. Available: https://searchWindowsserver.techtarget.com/tip/When-an-Active-Directory-design-goes-bad-and-how-to-fix-it.

[29] C. J. Udokwu, “ANALYSIS OF DIGITAL SECURITY THREATS IN AVIATION SECTOR,” Chibuzor Joseph Udokwu, Tallinn, 2017.

Appendix 1 – GroupsMonitoring.ps1

```
param (
   [Parameter(Mandatory=$true)][string]$zabbix_url="https://evdetect.cyber.net/ZBX/api/sender.php",
   [Parameter(Mandatory=$true)][string]$domain = "cyber.se",
   [Parameter(Mandatory=$true)][string]$zabbixhost = "InfraDir-ADcore-test-cyber"
)

function TrustCert{
add-type @"
using System.Net;
using System.Security.Cryptography.X509Certificates;
public class TrustAllCertsPolicy : ICertificatePolicy {
   public bool CheckValidationResult(
      ServicePoint srvPoint, X509Certificate certificate,
      WebRequest request, int certificateProblem) {
      return true;
   }
}
"@
   $AllProtocols = [System.Net.SecurityProtocolType]'Ssl3,Tls,Tls11,Tls12'
   [System.Net.ServicePointManager]::SecurityProtocol = $AllProtocols
   [System.Net.ServicePointManager]::CertificatePolicy = New-Object TrustAllCertsPolicy
}

if (-not ([System.Management.Automation.PSTypeName]'TrustAllCertsPolicy').Type)
{
   TrustCert
}

function send-zabbix {

   param(
      [Parameter(Mandatory=$true)]
      [string]$TargetHost,
      [Parameter(Mandatory=$true)]
      $data,
      [Parameter(Mandatory=$false)]
      [string]$Url
   )
   $body = ($data.Keys | % { @{host=$TargetHost; key=$_ ; value=$data[$_]} } | ConvertTo-Json)

   if (!$Url) {
      Write-Host "Zabbix API endpoint not specified...`n$($body | Out-String)"
      return
   }

   try {
      Invoke-WebRequest -Uri $Url -Method POST -Body $body
   } catch [exception] {
        $Host.UI.WriteErrorLine("ERROR: $($_.Exception.Message)")
    }
}


function getMember($groupName){
    $member = @(get-ADGroup -Identity $groupName -Properties * -server "cyber.se"| select -
ExpandProperty member)
    return ($member|sort) -join ", "
}

function domaincontroller{

    $groupsData =@(
        "Account Operators Test"
        "Administrators Test"
        "Backup Operators Test"
        "Domain Admins Test"
        "Enterprise Admins Test"
        "Print Operators Test"
        "Schema Admins Test"
        "Server Operators Test"
        "Allowed RODC Password Replication Group Test",
        "Certificate Service DCOM Access Test",
        "Cert Publishers Test",
        "Distributed COM Users Test",
        "DnsAdmins Test",
        "Event Log Readers Test",
        "Hyper-V Administrators Test",
        "Protected Users Test",
        "Remote Desktop Users Test",
        "WinRMRemoteWMIUsers__ Test"
    )

    return @{
        "account-operators-test" = getMember $groupsData[0]
        "administrators-test"= getMember $groupsData[1]
        "backup-operators-test" = getMember $groupsData[2]
        "domain-admins-test"= getMember $groupsData[3]
        "enterprise-admins-test"=getMember $groupsData[4]
        "print-operators-test"=getMember $groupsData[5]
        "schema-admins-test"=getMember $groupsData[6]
        "server-operators-test"=getMember $groupsData[7]
        "allowed-rodc-password-replication-group-test"=getMember $groupsData[8]
        "certificate-service-dcom-access-test"=getMember $groupsData[9]
        "cert-publishers-test"=getMember $groupsData[10]
        "distributed-com-users-test"=getMember $groupsData[11]
        "dnsadmins-test"=getMember $groupsData[12]
        "event-log-readers-test"=getMember $groupsData[13]
        "hyper-v-administrators-test"=getMember $groupsData[14]
        "protected-users-test"=getMember $groupsData[15]
        "remote-desktop-users-test"=getMember $groupsData[16]
        "winrmremotewmiusers__-test"=getMember $groupsData[17]
    }
}

function checkConnection{
    param([Parameter(Mandatory=$true)][String]$dc)

    $checkConnection = Test-Connection $dc -Quiet -Count 1

    return $checkConnection
}

if($(checkConnection -dc $domain) -eq $true){
    send-zabbix -TargetHost $zabbixhost -Data $(domaincontroller) -Url $zabbix_url
}
else{
    send-zabbix -TargetHost $zabbixhost -Data @{"Connection"=$(checkConnection)|Out-String} -Url
$zabbix_url
}
```

Appendix 2 – Tier0GroupsGathering.ps1

```
$dateTimeFile = (get-date).ToString("yyyy MM dd")
$outFile = $($PSScriptRoot)+"\Report-$($dateTimeFile).csv"
$outFileHTML =$($PSScriptRoot)+"\Report-$($dateTimeFile).html"
$delimiter = ","
$userName = Read-Host "Reporter"
$domain = Read-Host "Domain"
$groupsData =@(
        "Account Operators"
        "Administrators"
        "Backup Operators"
        "Domain Admins"
        "Enterprise Admins"
        "Print Operators"
        "Schema Admins"
        "Server Operators"
    )

$members = @()

foreach($group in $groupsData){
    $member = Get-ADGroupMember -Identity $group -Recursive -Server $domain |?{$_.objectClass -eq
"user"} | select -ExpandProperty sAMAccountName
    $members += $member
}

$userCount= $members.Count

$NeverExpires = 9223372036854775807

$global:ea = 0
$global:last2015 = 0
$global:last2016 = 0
$global:last2017 = 0
$global:otherLast = 0
$global:NeverLogon = 0
$global:noLastSet = 0
$global:passSet2015 = 0
$global:passSet2016 = 0
$global:passSet2017 = 0
$global:otherPassSet = 0
$global:accNotEx = 0
$global:accDisStatus=0
$global:smartRe =0
$global:passNExpSet = 0

foreach($member in $members){

    try{
```

```
    $memberInfor = Get-ADUser -Identity $member -Server $domain -Properties *|select
distinguishedName,
                                        sAMAccountName,
                                        mail,
                                        lastLogonTimeStamp,
                                        pwdLastSet,
                                        accountExpires,
                                        userAccountControl,
                                        Description

    #Last Logon
    $lastLogon = [datetime]::fromfiletime($memberInfor.lastLogonTimeStam)
    $lastLogon= $lastLogon.ToString("yyyy/MM/dd")
    if($lastLogon.split("/")[0] -eq 2015){
      $global:last2015++
    }
    elseif ($lastLogon.split("/")[0] -eq 2016){
      $global:last2016++
    }
    elseif ($lastLogon.split("/")[0] -eq 2017){
      $global:last2017++
    }elseif ($lastLogon.split("/")[0] -eq 1601){
      $lastLogon = "Never"
      $global:NeverLogon++
    }else{
      $global:otherLast++
    }

    #password last set
    if($memberInfor.pwdLastSet -eq 0)
    {
       $pwdLastSet = "Never"
       $global:noLastSet++
    }
    else
    {
       $pwdLastSet = [datetime]::fromfiletime($memberInfor.pwdLastSet)
       $pwdLastSet = $pwdLastSet.ToString("yyyy/MM/dd")
       if($pwdLastSet.split("/")[0] -eq 2015){
          $global:passSet2015++
       }
       elseif ($pwdLastSet.split("/")[0] -eq 2016){
          $global:passSet2016++
       }
       elseif ($pwdLastSet.split("/")[0] -eq 2017){
          $global:passSet2017++
       }
       elseif ($pwdLastSet.split("/")[0] -eq 1601){
          $pwdLastSet = "Never"
          $global:noLastSet++
       }
       else{
            $global:otherPassSet++
        }

    }
    #Account expires
    if(($memberInfor.accountExpires -eq $NeverExpires) -or ($memberInfor.accountExpire -gt
[Datetime]::MaxValue.Ticks))
    {
        $convertAccountEx = "Not Expired"

    }
    else
    {
        $convertAccountEx = "Expired"
        $global:accEx++
    }
    #Email
    if([String]::IsNullOrEmpty($mail)){
        $email = "N/A"
    }
    else{
        $email =$mail
        $global:ea++
    }

    #UserInfor
    if($memberInfor.userAccountControl -band 0x0002)
    {
        $accountDisStatus = "disabled"
        $global:accDisStatus++
    }
    else
    {
        $accountDisStatus = "enabled"
    }
    #If Smartcard Required
    if( $memberInfor.userAccountControl -band 262144)
    {
        $smartCDStatus = "Required"
        $global:smartRe++
    }
    else
    {
        $smartCDStatus = "Not Required"
    }

    #If No password is required
    if( $memberInfor.userAccountControl -band 32){
        $passwordEnforced ="Not Required"
        $global:passNotRe++
    }
    else
```

```
        {
           $passwordEnforced = "Required"
        }

        #Password never expired
        if( $memberInfor.userAccountControl -band 0x10000){
           $passNExp ="Never Expires is set"
           $global:passNExpSet++

        }
        else
        {
           $passNExp = "None Set"
           $passTrue = $true
        }

        $obj = New-object -TypeName psobject
        $obj | Add-Member -MemberType NoteProperty -Name "Distinguished Name" -Value
$memberInfor.distinguishedName
        $obj | Add-Member -MemberType NoteProperty -Name "Sam account" -Value
$memberInfor.sAMAccountName
        $obj | Add-Member -MemberType NoteProperty -Name "Email" -Value $memberInfor.mail
        $obj | Add-Member -MemberType NoteProperty -Name "Password last changed" -Value
$pwdLastSet
        $obj | Add-Member -MemberType NoteProperty -Name "Last Logon " -Value $lastLogon
        $obj | Add-Member -MemberType NoteProperty -Name "Account Expires" -Value
$convertAccountEx
        $obj | Add-Member -MemberType NoteProperty -Name "Account Status" -Value $accountDisStatus
        $obj | Add-Member -MemberType NoteProperty -Name "Smartcard Required" -Value
$smartCDStatus
        $obj | Add-Member -MemberType NoteProperty -Name "Password Required" -Value
$passwordEnforced
        $obj | Add-Member -MemberType NoteProperty -Name "Never Expired Password Set" -Value
$passNExp
        $obj | Add-Member -MemberType NoteProperty -Name "Description" -Value
$memberInfor.Description
        #$obj
        $obj | Export-Csv -Path "$outFile" -NoTypeInformation -append -Delimiter $delimiter
    }
    catch{

        Write-Error "Can't get this user $member"
    }



}
# Saving images to import later to the HTML reports
$global:IncludeImages = New-Object System.Collections.ArrayList
$global:check= 0
$global:outFilePicPie = $($PSScriptRoot)+"\Pie-$($dateTimeFile)-$($global:check).jpeg"
#PIE
#Email
```

```
$emailPer = $global:ea
$noEmailPer=  $userCount - $emailPer
$mailHash = @{"Available"=$emailPer;"Unavailable"=$noEmailPer}

#Account expired
$accExPer = $global:accEx
$accNotExPer = $userCount - $accExPer
$accExHash = @{"Expired"="$accExPer";"Unexpired"="$accNotExPer"}

#Account Status
$accDisPer = $global:accDisStatus
$accNoDisPer = $userCount - $accDisPer
$accStatusHash = @{"Disabled"="$accDisPer";"Enabled"="$accNoDisPer"}

#Smart Card required
$smartRePer = $global:smartRe
$smartNotRePer = $userCount - $smartRePer
$smartReHash = @{"Required"="$smartRePer";"Not Required"="$smartNotRePer"}

#Password Required
$passReNotPer = $global:passNotRe
$passRePer =  $userCount - $passReNotPer
$passReHash = @{"Not Required"="$passReNotPer";"Required"="$passRePer"}

#Password Never Expired Set
$passExpSetPer =$global:passNExpSet
$passExpNoSetPer= $userCount - $passExpSetPer
$passExpHash = @{"Set"="$passExpSetPer";"None-set"="$passExpNoSetPer"}
Function drawPie {
   param($hash,
   [string]$title
   )
   Add-Type -AssemblyName System.Windows.Forms
   Add-Type -AssemblyName System.Windows.Forms.DataVisualization
   $Chart = New-object System.Windows.Forms.DataVisualization.Charting.Chart
   $ChartArea = New-Object System.Windows.Forms.DataVisualization.Charting.ChartArea
   $Series = New-Object -TypeName System.Windows.Forms.DataVisualization.Charting.Series
   $ChartTypes = [System.Windows.Forms.DataVisualization.Charting.SeriesChartType]
   $Series.ChartType = $ChartTypes::Pie
   $Chart.Series.Add($Series)
   $Chart.ChartAreas.Add($ChartArea)
   $Chart.Series['Series1'].Points.DataBindXY($hash.keys, $hash.values)
   $Chart.Series[‘Series1’][‘PieLabelStyle’] = ‘Disabled’
   $Legend = New-Object System.Windows.Forms.DataVisualization.Charting.Legend
   $Legend.IsEquallySpacedItems = $True
   $Legend.BorderColor = 'Black'
   $Chart.Legends.Add($Legend)
   $chart.Series["Series1"].LegendText = "#VALX (#VALY)"
   $Chart.Width = 700
   $Chart.Height = 400
   $Chart.Left = 10
   $Chart.Top = 10
   $Chart.BackColor = [System.Drawing.Color]::White
   $Chart.BorderColor = 'Black'
   $Chart.BorderDashStyle = 'Solid'
   $ChartTitle = New-Object System.Windows.Forms.DataVisualization.Charting.Title
   $ChartTitle.Text = $title
   $Font = New-Object System.Drawing.Font @('Microsoft Sans Serif','12',
[System.Drawing.FontStyle]::Bold)
   $ChartTitle.Font =$Font
   $Chart.Titles.Add($ChartTitle)
   $testPath = Test-Path $global:outFilePicPie
   if($testPath -eq $True){
      $global:check += 1
      $global:outFilePicPie = $($PSScriptRoot)+"\Pie-$($dateTimeFile)-$($global:check).jpeg"
   }
   $global:IncludeImages.Add($global:outFilePicPie)
   $Chart.SaveImage($outFilePicPie, 'jpeg')
}
#BAR
#lastLogon
$lastLogonHash =
[ordered]@{"Never"="$global:NeverLogon";"<2015"="$global:otherLast";"2015"="$global:last2015";"2
016"="$global:last2016";"2017"="$global:last2017"}
$global:check1= 0
$global:outFilePicBar = $($PSScriptRoot)+"\Bar-$($dateTimeFile)-$($global:check).jpeg"

#PassLastSet
$passSetHash =
[ordered]@{"Never"="$global:noLastSet";"<2015"="$global:otherPassSet";"2015"="$global:passSet201
5";
            "2016"="$global:passSet2016";"2017"="$global:passSet2017";}

function drawBar{
   param(
   $hash,[string]$title,[string]$axisX
   )
   Add-Type -AssemblyName System.Windows.Forms
   Add-Type -AssemblyName System.Windows.Forms.DataVisualization
   $Chart1 = New-object System.Windows.Forms.DataVisualization.Charting.Chart
   $ChartArea1 = New-Object System.Windows.Forms.DataVisualization.Charting.ChartArea
   $Series1 = New-Object -TypeName System.Windows.Forms.DataVisualization.Charting.Series
   $ChartTypes1 = [System.Windows.Forms.DataVisualization.Charting.SeriesChartType]
   $Chart1.Series.Add($Series1)
   $Chart1.ChartAreas.Add($ChartArea1)
   $Chart1.Series['Series1'].Points.DataBindXY($hash.keys, $hash.values)
   $chart1.Series[0].ChartType =
[System.Windows.Forms.DataVisualization.Charting.SeriesChartType]::Column
   $ChartArea1.AxisX.Title = $axisX
   $ChartArea1.AxisY.Title = "Figures"
   $Chart1.Series['Series1'].IsValueShownAsLabel = $True
   $Chart1.Series['Series1'].SmartLabelStyle.Enabled = $True
   $chart1.Series['Series1']["LabelStyle"] = "TopLeft"
```

```
    $ChartArea1.AxisY.Maximum = $userCount

    if($userCount -ge 1000){
        $ChartArea1.AxisY.Interval = $inter - ($inter %100)
        $inter = [math]::Round($userCount/10,0)
    }elseif($userCount -ge 100){
        $ChartArea1.AxisY.Interval = $inter - ($inter %10)
        $inter = [math]::Round($userCount/20,0)
    }else{
        $ChartArea1.AxisY.Interval = $inter - ($inter %10)
        $inter = [math]::Round($userCount/10,0)
    }

    $Chart1.Width = 1000
    $Chart1.Height = 700
    $Chart1.Left = 10
    $Chart1.Top = 10
    $Chart1.BackColor = [System.Drawing.Color]::White
    $Chart1.BorderColor = 'Black'
    $Chart1.BorderDashStyle = 'Solid'
    $ChartTitle1 = New-Object System.Windows.Forms.DataVisualization.Charting.Title
    $ChartTitle1.Text = $title
    $Font1 = New-Object System.Drawing.Font @('Microsoft Sans Serif','12',
[System.Drawing.FontStyle]::Bold)
    $ChartTitle1.Font =$Font1
    $Chart1.Titles.Add($ChartTitle1)

    $testPath = Test-Path $global:outFilePicBar
    if($testPath -eq $True){
        $global:check1 += 1
        $global:outFilePicBar = $($PSScriptRoot)+"\Bar-$($dateTimeFile)-$($global:check1).jpeg"
    }
    $global:IncludeImages.Add($global:outFilePicBar)
    $Chart1.SaveImage("$outFilePicBar", 'jpeg')
}
# Draw Pie
drawPie -hash $mailHash -title "Emails" |Out-Null
drawPie -hash $accExHash -title "Expired Accounts"|Out-Null
drawPie -hash $accStatusHash -title "Account Status"|Out-Null
drawPie -hash $smartReHash -title "Smart Cards Required"|Out-Null
drawPie -hash $passReHash -title "Password Required"|Out-Null
drawPie -hash $passExpHash -title "Password Never Expired Settings"|Out-Null

# Draw bar
drawBar -hash $lastLogonHash -title  "Last Logon Date" -axisX "years"|Out-Null
drawBar -hash $passSetHash -title "Password Last Changed" -axisX "years"|Out-Null

# Generating HTML Reports
$groupsData = $groupsData|Out-String
$body =@'
<h1> Forest Report </h1>
<p><ins><b>I.<b> Information<ins></p>
```

```
<div class="tablehere">
   <table class="tabinfo" >
        <tr>
          <td>Reported by:</td>
          <td>{0}</td>
        </tr>
        <tr>
          <td>Datetime:</td>
          <td>{1}</td>
        </tr>
   </table>
</div>
<div class="tabofexecu">
   <table class="tabexecu" >
        <tr>
          <td>List of tier0 groups:</td>
          <td>{2}</td>
        </tr>
        <tr>
          <td>Number of accounts: </td>
          <td>{3}</td>
        </tr>

   </table>
<div>

<p><ins><b>III.<b> Data Illustration<ins></p>
'@ -f  $userName,$(get-date),$groupsData,$members.Count

function Generate-Html {
   Param(
      [Parameter()]
      [string[]]$IncludeImages
   )

   if ($IncludeImages){
      $ImageHTML = $IncludeImages | % {
      $ImageBits = [Convert]::ToBase64String((Get-Content $_ -Encoding Byte))
      "<center><img src=data:image/jpeg;base64,$($ImageBits) alt='My Image'/><center>"
   }
      ConvertTo-Html -Body $body -PreContent $imageHTML -Title "Report on $Domain" -CssUri
"Style.css" |
      Out-File $outFileHTML
   }
}

Generate-Html -IncludeImages $global:IncludeImages

foreach($image in $IncludeImages){

   rm $image
}
```

```
#Finish
Write-Host "`nData file has been save in $outFile" -ForegroundColor Cyan
Write-Host "`nReport file has been save in $outFileHTML`n" -ForegroundColor Cyan
Write-Verbose -Message  "Script Finished!!" -Verbose
```

Appendix 3 – Style.css

```
img{

display: block;
   margin-left: auto;
   margin-right: auto;
margin-bottom: 20px;
margin-top: 20px;
}
body {
            background-color: #b7c8db;

}

table.tabinfo th, td {
            border: 1px solid black;
            table-layout: fixed;
}
tabe.tabexecu th,td{
            border: 1px solid black;
            table-layout: fixed;
            white-space:pre;
}
.tabinfo{
            background-color: white;
            margin-bottom: 20px;
            margin-top: 20p;
            margin-left: 40px;
            height:100%;
            width:auto;

}

.tabexecu{
            background-color: white;
            margin-bottom: 20px;
            margin-top: 20p;
            margin-left: 40px;
            widthl:auto;
            height:100%;

}
```