F-Secure

TENDING TO THE RED FOREST

Considerations and harsh realities of a Red Forest implementation

An F-Secure Consulting whitepaper
written by Christo Erasmus
and Christopher Panayi

CONTENTS

1 SUMMARY

Active Directory (AD) is used as the main enterprise single sign-on (SSO) solution in most large corporate networks. It's an extremely attractive target for attackers, since having control over an organization's AD would typically result in having control over the entire network.

What problem are we trying to solve?

A significant challenge faced by many large organizations is adequately securing their AD environment. The security posture of an AD domain relies heavily on configuration and architecture. Resolving vulnerabilities in a domain isn't simply a matter of applying patches - it primarily requires security controls that would cause a change in IT administration processes, ultimately leading to business impact.

What's the perfect fix?

In general terms, the way to solve this problem is to **make it impossible for an account or system (or even an unauthenticated entity) to get into a position to compromise a more privileged account**. The real difficulty with implementation is introduced when organizations attempt to design, consider all the relevant aspects, and implement this solution without significantly disrupting business operations. This issue was so ubiquitous Microsoft developed a framework to implement the solution: The Enhanced Security Administrative Environment (ESAE), also known as "Red Forest".

Expectations vs. reality

At its core, ESAE is based on a tiered administration model for Active Directory, where accounts and systems are divided into separate tiers, each corresponding to different levels of privilege. This is supplemented by additional controls, designs and suggested processes, in order to support the tier model, or to prevent other types of attacks that could undermine it.

In reality, it may not be practical for most organizations to implement the full Red Forest model - at least not within a reasonable time frame. Some aspects of the model are understandably difficult to implement in a large, complex environment with many legacy issues. Other aspects may impact operations to such an extent, the organization's unwilling to implement them.

> While a complete implementation of Red Forest has clear benefits for security, the extent to which an incomplete implementation would improve AD security is far less certain.

Some aspects of the model don't provide a significant benefit in isolation. In addition, not all aspects would provide the same level of improvement in AD security, depending on the overall security posture of the organization. For this reason, many incomplete Red Forest implementations are unlikely to meet the expectation of making AD secure.

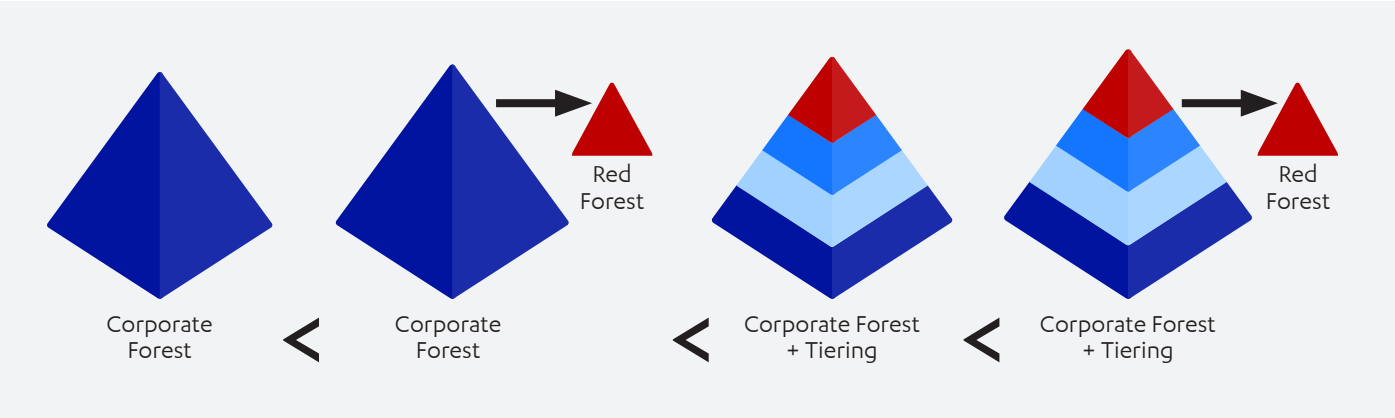
Reasons why some incomplete implementations may not solve the problem

- Some aspects of the Red Forest model do not provide a significant benefit in isolation.
- Not all aspects would provide the same level of improvement in AD security, depending on the overall security posture of the organization.
- Some controls can end up being almost completely ineffective if they're not implemented correctly. Understanding the attack that a control is meant to prevent would allow for analysis of whether a control would still work as intended if any changes or deviations are made.

An important example of these principles is that, despite a complete implementation of AD tiering requiring a significant amount of time and effort to complete, other aspects of the Red Forest model would simply not be effective without it.

Even a separate administration forest, which does provide limited benefits in isolation, would not significantly improve AD security without a properly implemented tier model. In comparison, a single corporate forest with a tiering implementation and no separate administration forest would be more effective at preventing attacks. This example is illustrated in Figure 1.

Figure 1: *Comparison between effectiveness of different combinations of tiering and separate administration forests*



Recommended prioritization of critical ESAE components

A few examples of incomplete implementations are presented in Section 3, along with attack scenarios illustrating how these implementations could still be compromised. The point of these examples is ultimately to determine what would be required for a Red Forest implementation to be successful. One of the key points these examples illustrate, is that **a complete tiering implementation is absolutely integral to the Red Forest model**. Without it, many of the other controls wouldn't be nearly as effective as they should be. There are also a few specific controls that could significantly improve a domain or forest's security posture, without requiring a lot of effort to implement or significantly disrupting operations.

These components should ideally be prioritized in a Red Forest implementation:

- Implement a complete and comprehensive tier model
- Implement privileged access workstations (PAWs) for Tier 0
- Use Restricted Admin mode for RDP on all suitable machines
- Address service accounts with excessive privileges and weak passwords
- Identify and remediate misconfigurations that could be used to compromise privileged accounts
- Enable the Windows Firewall on workstations and restrict access to remote administration services (e.g. SMB and RDP)
- Secure local account management
- Implement a PAM solution

This list is by no means a comprehensive set of controls for securing AD, but it is a good starting point. Although some of these components would likely be complex and time-consuming to implement, a Red Forest implementation would be ineffective without them.

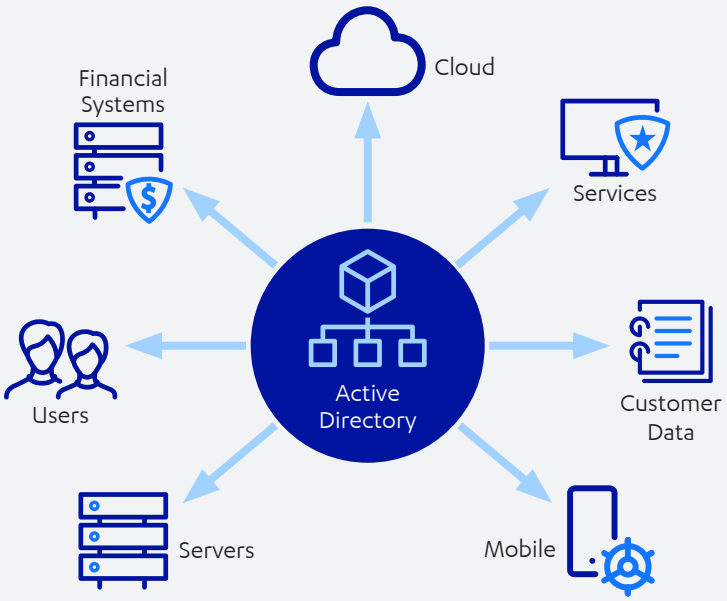
If an organization does implement these components successfully, it would make their AD environment significantly harder to compromise and increase their overall resilience towards targeted attacks. If desired, the organization would then be primed to commence with a full Red Forest implementation to secure the environment even further.

2 BACKGROUND

In most large corporate networks, a Windows Active Directory (AD) environment is the main enterprise-wide single sign-on (SSO) solution and underpins virtually all critical systems, as shown in Figure 2 If AD is compromised, it would directly lead to the compromise of these other systems as well.

The users and administrators for these systems typically make use of AD accounts and operate from workstations joined to an AD domain. Some of these systems might also use AD as their authentication provider. Control over AD would therefore mean control over the system, its users, and its administrators. AD is an extremely attractive target for attackers, as it could allow them to gain a significant level of control over an environment, often with relative ease.

Figure 2: *Active Directory's central role in an enterprise network*



2.1 What problem are we trying to solve?

A significant challenge faced by many large organizations is adequately securing their AD environment. The security posture of an AD domain relies heavily on configuration and architecture. Resolving vulnerabilities in a domain isn't simply a matter of applying patches - it primarily requires security controls that would cause a change in IT administration processes, ultimately leading to business impact.

Large, complex AD environments, some built more than a decade ago, are often affected by a variety of systemic security vulnerabilities that aren't easy to resolve without impacting business operations. Even with the realization that AD security's critically important, it can be difficult to determine **how** to secure an AD environment practically, or where to start. In other cases, a significant amount of effort may have already been invested into securing AD, only to discover it can still be compromised with relative ease during a penetration test.

In the context of an enterprise security strategy, it's widely recognized that, in general, prevention controls are not perfect and shouldn't be relied upon in isolation. However, their role in slowing attackers down enough for detection and response controls to be effective is important.

If an attacker's able to compromise an organization's AD in a matter of hours, even a highly capable blue team would likely have difficulty fending off the attack before highly privileged credentials are compromised. On the other hand, an AD environment with strong security controls could severely restrict an attacker's options once they land on the network. This provides defenders with more time to detect them, or forces attackers to resort to less stealthy techniques. This could significantly increase an organization's chances of detecting an attack during the early stages, when it's generally far simpler to contain the attack and eradicate the attacker's presence on the network.

How does this happen?

If our aim is to ensure that Active Directory is secure against compromise, it may be useful to investigate the anatomy of a typical Active Directory attack. Most AD compromises are the result of an attacker performing privilege escalation. The starting point for this process might be having control over a standard employee's domain account after a successful phishing attack, or access to a particular system through a supply chain attack. Through whatever means, these low-privileged accounts or systems would then be used to compromise accounts or systems with more privileges. These could

have access to more systems, which could be used to compromise additional accounts with even more privileges. This process would be repeated until an account or system with sufficient privileges to assume control over the domain is compromised.

Depending on the security posture of the environment in question, this process could consist of only a single step, or it could require many steps and a variety of different vulnerabilities. Some of the main types of issues that make privilege escalation possible in AD are:

- **Excessive privileges**: If accounts are assigned more privileges than they require, it would increase the impact if they are compromised.
- **Lack of privilege separation**: If high-privileged accounts log in to hosts where low-privileged accounts have administrative rights, the low-privileged accounts would be in a position to compromise the high-privileged accounts. Privilege escalation opportunities could arise when accounts of different privilege levels make use of the same systems.
- **Weak passwords**: A simple password guessing attack could be an effective way to compromise highly privileged accounts. For example, attacks such as Kerberoasting[1] or AS-REP Roasting[2] could be particularly effective, as they can be performed from a low-privileged context and allow for offline password guessing.
- **Lack of hardening controls**: This ranges from controls that reduce the amount of credential material stored in memory on Windows hosts, to network-based controls, such as enforcing SMB Signing or disabling LLMNR and NBTNS. A lack of these controls could make certain attacks possible, and others easier to perform.
- **Misconfigurations and vulnerabilities**: Misconfigurations in AD itself may have unintended consequences and lead to privilege escalation opportunities. This could also be the case for enterprise software integrated closely with AD - any weaknesses in these solutions could be exploited by attackers, who could then abuse the solutions' privileges in AD.

2.2 A typical attack

To provide a tangible example of how an attacker could perform such an attack, a simple attack scenario is presented. This scenario will later serve as a basis for evaluating the effectiveness of proposed hardening controls for Active Directory. The scenario takes place in the network of a fictional organization, ACME, that has an Active Directory domain called **acme.corp**.

The starting point for the attack is a standard low-privileged domain user account, Bob, located in the **acme.corp** domain. The attacker gained control over this account by compromising Bob's workstation through a phishing attack. Using Bob's account, the attacker could enumerate the domain, using tools such as PowerView or BloodHound. This would provide the attacker with detailed information, including the following:

- Details of all users and machines
- Domain group memberships
- Members of the local Administrators group on all domain-joined machines
- Active logged-in sessions for domain users
- ACLs on domain objects
- Group Policy

Using this information, the attacker can identify potential privilege escalation paths in the acme.corp domain. They can also determine which privileges have been assigned to Bob's user account, and whether they could use it to compromise any additional users or systems.

[1] Refer to [1], https://adsecurity.org/?p=2293
[2] Refer to [2], https://www.harmj0y.net/blog/activedirectory/roasting-as-reps/

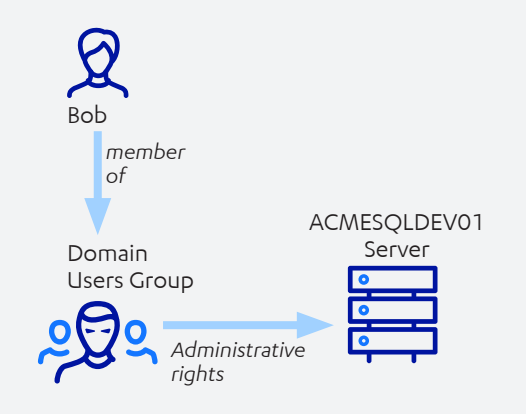


Figure 3: *A typical attack against AD - step 1*

STEP 1

The attacker observes that the "Domain Users" group had been added to the local Administrators group on several machines. As a result, even users with no intentional administration privileges would be able to assume control over these machines – Bob's account being one such user. The attacker uses Bob's account to deploy a payload over SMB to one of these machines - ACMESQLDEV01, as shown in Figure 3.

STEP 2

Now the attacker has administrative access to ACMESQLDEV01, they can extract cached credentials from the host's memory. This would provide the attacker with password hashes or cleartext passwords for accounts with active sessions (i.e. users actively logged in), or accounts configured to run services on the host.

Using this technique, the password hash is obtained for the *svc_sqldev account, a member of the "ACME SQL Cluster Admin" group. This* group is a member of the local Administrators group on all of the SQL servers related to the ACME application, including the production servers, as shown in Figure 4.

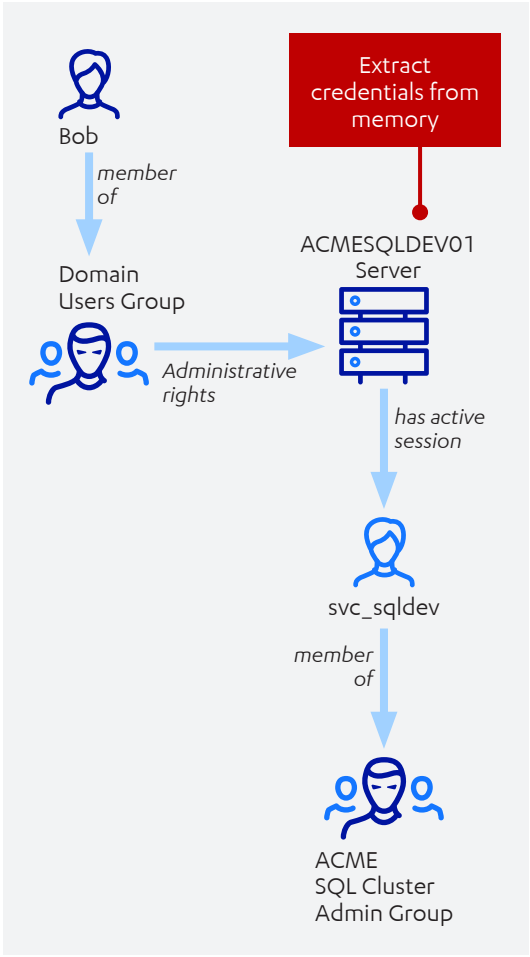


Figure 4: *A typical attack against AD - step 2*

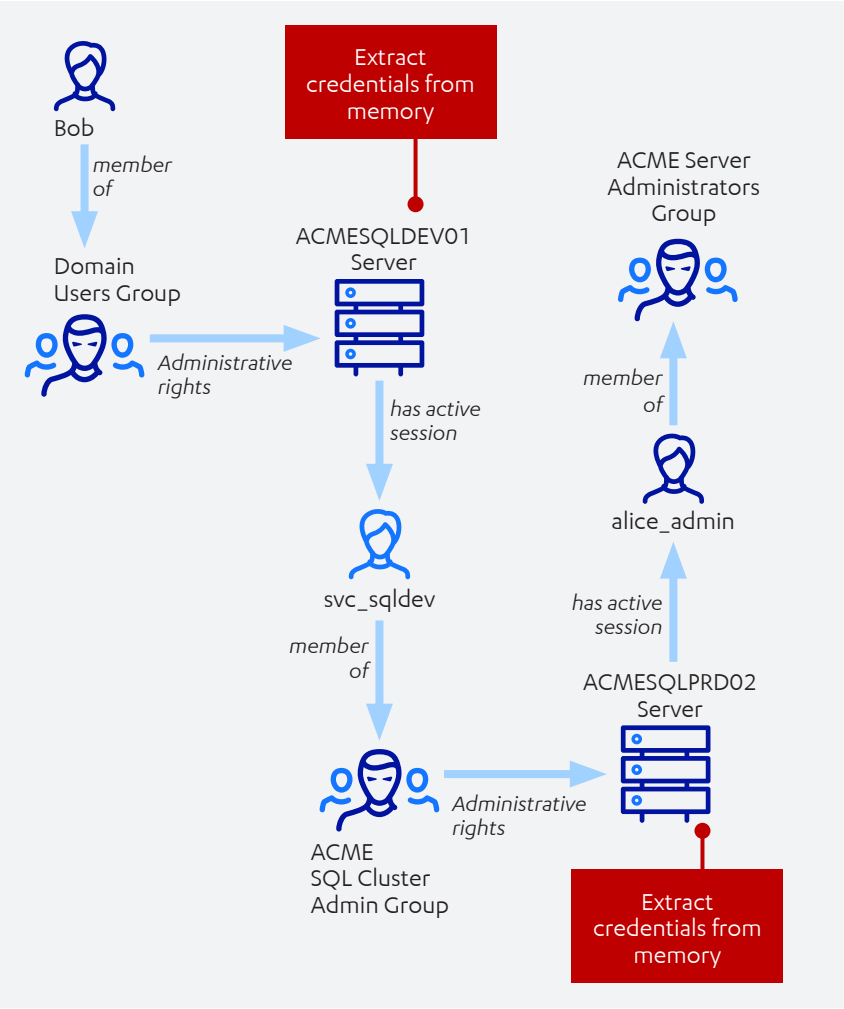


Figure 5: *A typical attack against AD - step 3*

STEP 3

The attacker sees a privileged server administration account is logged on to one of the production SQL servers, ACMESQLPRD02. Through a similar method as before, the attacker uses the *svc_sqldev* account's password hash, exploiting the Pass-the-Hash nature of Windows authentication. They deploy a payload on the ACMESQLPRD02 server, and extract credentials from memory.

The password hash for the privileged account, *alice_admin*, is successfully obtained. The account's a member of the highly privileged "ACME Server Administrators" domain group, which has administrative privileges on the majority of the server infrastructure in the **acme.corp** domain. This is illustrated in Figure 5.

STEP 4

The attacker could now use the *alice_admin* account to compromise almost any server they desire. The next goal is to compromise a member of the "Domain Admins" group, or any other system or account with similar privileges.

The attacker decides to target environments where domain administrators are commonly active, such as Microsoft Exchange. Using the techniques described before, they deploy payloads to these servers and extract credentials from memory. On one of the servers, a member of the "Domain Admins" group was logged in, *john_admin*. The attacker now has their password hash, and can use it to compromise a domain controller, or retrieve password hashes over the network through a DCSync attack[3], as shown in Figure 6.

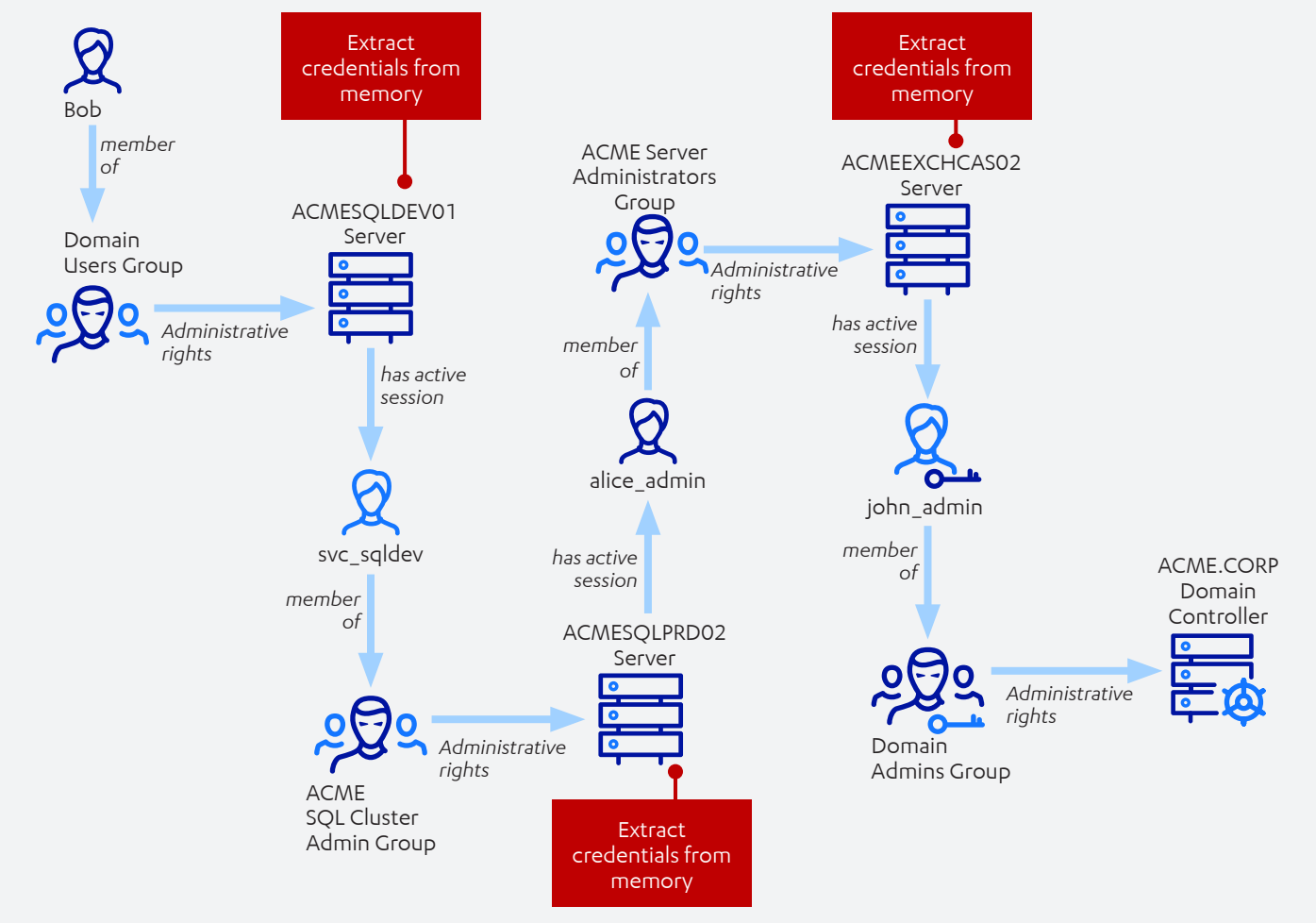


Figure 6: *A typical attack against AD - step 4*

At this point the **acme.corp** domain is considered completely compromised. The attacker is now able to retrieve the password hashes for all domain user accounts, which essentially grants them the ability to access any system or impersonate any user in the domain.

2.3 What's the perfect fix?

In general terms, the way to solve this problem is to **make it impossible for an account or system (or even an unauthenticated entity) to get into a position to compromise a more privileged account**. This relates most directly to the concept of privilege separation - where accounts of different privilege levels aren't permitted to make use of the same systems, and the accounts only possess absolutely necessary privileges. It also applies to cases where misconfigurations or vulnerabilities could be exploited by relatively low-privileged users, in order to compromise more privileged users or systems.

Although this may be an overly simplistic statement, the fundamental principle is important. The real difficulty with implementation is introduced when organizations attempt to design, consider all the relevant aspects, and implement this solution without significantly disrupting business operations. This issue was so ubiquitous Microsoft developed a framework to implement the solution: The Enhanced Security Administrative Environment (ESAE), also known as "Red Forest". This provides organizations with a detailed model which states whether a new AD environment can be built, or an existing environment can be secured.

[3] A feature in Mimikatz, written by Benjamin Delpy and Vincent Le Toux. Refer to [3], https://adsecurity.org/?p=1729

2.4 Defining Red Forest

Red Forest isn't the only correct solution, but it is comprehensive, detailed, well-defined, and has become quite well-known. For this reason, it's a good reference point for discussions regarding AD security, even in cases where only certain aspects of the model might be used.

ESAE covers a broad range of controls and a full discussion of the model is beyond the scope of this paper. For the purpose of this discussion, focus will be placed on defining an effective implementation of Red Forest, specifically with regards to defending against the types of attacks discussed above.

At its core, ESAE is based on a tiered administration model for Active Directory, where accounts and systems are divided into separate tiers, each corresponding to different levels of privilege. This is supplemented by additional controls, designs, and suggested processes – either to support the tier model, or to prevent other types of attacks that could undermine it.

Red Forest implements the following features which are especially relevant to this discussion:

- **Privilege separation (tiering)**: Accounts and systems are divided into tiers, based on their level of privilege. Accounts within a tier don't have any administrative rights over systems in higher tiers, and aren't generally permitted to authenticate to systems in a lower tier. The purpose of this model is primarily to prevent privilege escalation through credential theft attacks, since accounts of different privilege levels wouldn't have access to the same systems.
- **Separation of administration systems**: The hosts from where administration is performed, e.g. administrators' workstations, are separated and secured according to the tier model. These systems should not be accessible by accounts or systems from lower tiers, to protect their users from credential theft attacks. The Red Forest model utilizes the concept of a Privileged Access Workstation (PAW)[4] that is used for this purpose.
- **Separate administration forests**: Separate administration forests need to be created for administrative accounts and systems that form part of the tier model. For example, an "Admin" forest for AD administration activities (Tier 0), and a "PRIV" forest configured with a Privileged Account Management (PAM) solution for other administration accounts (Tier 1 and 2). These forests should be trusted by the corporate forest, but the administration forests shouldn't trust the corporate forest. Configuring separate administration forests would help reduce the attack surface of the administrative users and systems. If an organization's AD has been compromised and needs to be rebuilt, starting with a separate administration forest would help ensure the new AD couldn't be compromised through the existing forest.
- **Hardening controls**: The Red Forest model recommends the implementation of a large number of different hardening controls. For example, some recommended controls are aimed at increasing the difficulty of performing credential theft attacks, such as Credential Guard and the Protected Users domain group. Some of these controls are aimed at making certain attacks more difficult or limiting their impact, while others make some types of attacks impossible to perform.
- **Privileged Account Management (PAM)**: According to the tier model, administrators would have separate accounts for administration activities and normal business activities. Some administrators may have several different accounts, and it could become difficult for them to set and remember several different passwords for all their accounts. A PAM solution would assist these users with managing their accounts and provide a mechanism to make sure all privileged accounts have strong passwords that can be maintained and monitored on an ongoing basis. The Red Forest model also includes more advanced PAM concepts, such as Just Enough Administration (JEA) and Just In Time Administration (JIT)[5], which allow for more fine-grained control over privileged accounts.

[4] Refer to [4], https://docs.microsoft.com/en-us/windows-server/identity/securing-privileged-access/privileged-access-workstations

[5] Refer to [5], https://docs.microsoft.com/en-us/microsoft-identity-manager/pam/privileged-identity-management-for-active-directory-domain-services

2.5 Expectations vs. reality

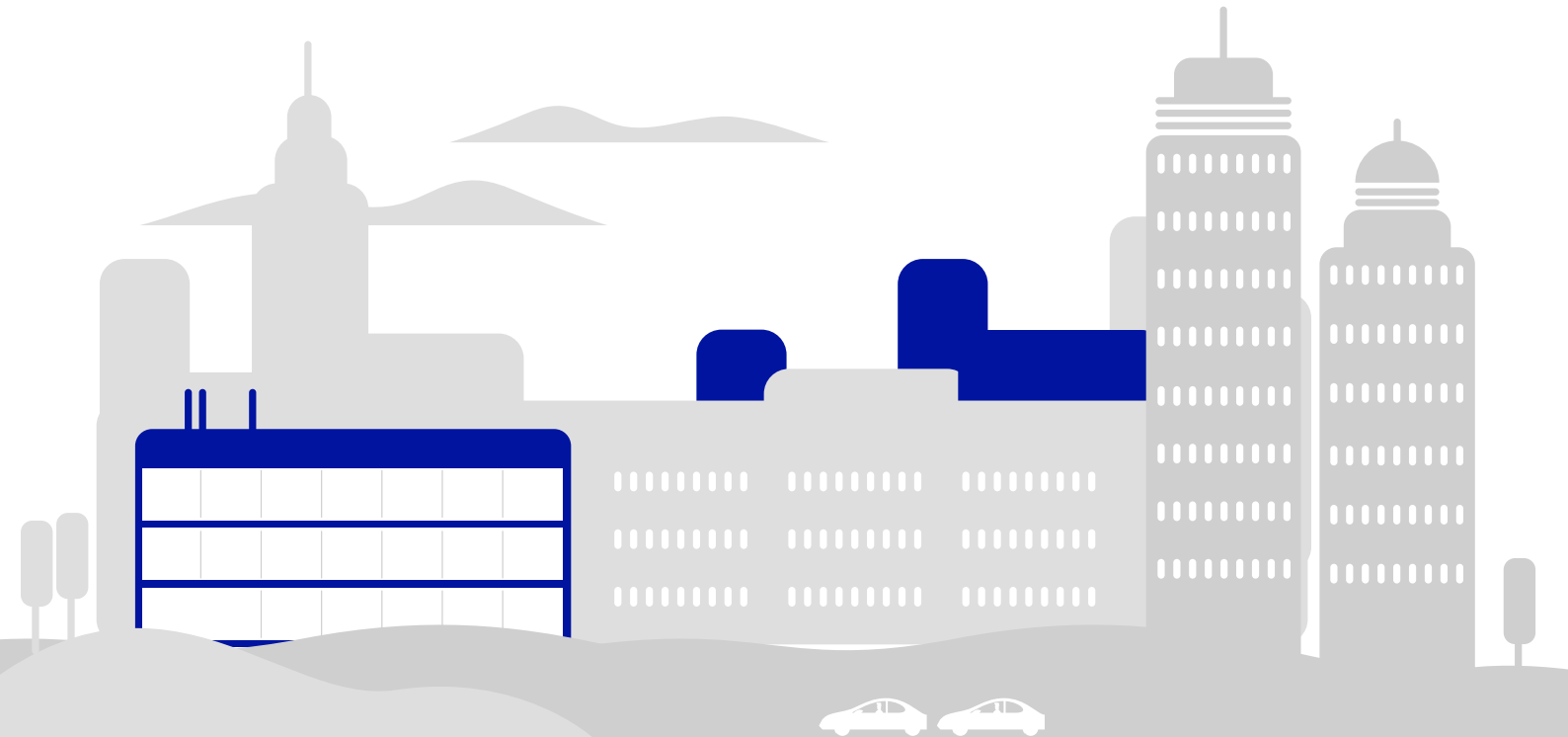
The understandable expectation of those looking at the ESAE model is that implementing Red Forest will make an AD environment secure. This expectation isn't necessarily unrealistic. A complete and correct implementation of ESAE would result in a nearly impenetrable AD environment - at least when considering the vast majority of AD attacks known today. However, when it comes to practical implementations of ESAE in real-world environments, matters can become more complicated.

The term "Red Forest" can mean different things to different people. An implementation that consists primarily of the "Separate administration forests", but doesn't properly consider the other aspects of the model, would ultimately be ineffective. In reality, it may not be practical for most organizations to implement the full Red Forest model - at least not within a reasonable time frame. Some aspects of the model are understandably difficult to implement in a large, complex environment with many legacy issues. Other aspects may impact operations to such an extent, the organization's unwilling to implement them.

As a result, organizations would most likely end up with an incomplete implementation of Red Forest, or would be busy with their implementation for the foreseeable future.

> While a complete implementation of Red Forest has clear benefits for security, the extent to which an incomplete implementation would improve AD security is far less certain.

Some aspects of the model don't provide a significant benefit in isolation. In addition, not all aspects would provide the same level of improvement in AD security, depending on the overall security posture of the organization. For this reason, many incomplete Red Forest implementations are unlikely to meet the expectation of making AD secure. The remainder of this discussion explores the reasons why some incomplete implementations wouldn't be effective, and provides suggestions regarding which aspects of the Red Forest model should be prioritized for implementation.



3 EXAMPLES OF INCOMPLETE IMPLEMENTATIONS

This section presents three examples of incomplete implementations, partly based on issues observed in real environments during security assessments.

3.1 Enterprise software

In the first example, the organization has taken the following actions as part of their Red Forest implementation:

1. Built a separate administration forest, **acme.admin**
2. Implemented tiering for employee accounts
3. Administrators' tiered accounts were created in the administration forest

Although tiering was correctly implemented for all the employees performing administration activities, some highly privileged enterprise software solutions were still in place in the corporate forest, **acme.corp**. The organization made use of an enterprise backup solution, which had a service account that required administrative rights on all the machines in the domain to perform backups - including on domain controllers. This service account, *svc_backup*, was a member of the "Administrators" domain group, which has similar privileges to the "Domain Admins" group.

The *svc_backup* account was still present in the corporate forest, along with the servers where the backup solution (ACMEBACKUP01) was installed. The account was considered secure, with a strong, random password that was changed regularly. However, the organization failed to consider the fact the account's credentials were cached on the server where the backup solution was installed. As a result, any user account with administrative access to the ACMEBACKUP01 server would be able to obtain the backup service account's credentials.

AD Compromise Path

In this scenario, the backup server wasn't recognized as a Tier 0 system, despite it possessing Tier 0 privileges. Several Tier 1 domain groups had administrative access to this server, including "ACME Server Administrators" and "ACME Backup Admins". If any of the accounts in these groups were compromised, they could be used to access the ACMEBACKUP01 server and obtain credentials for the *svc_backup* account. These credentials could then be used to compromise Tier 0 in the corporate forest, at which point the entire corporate forest would be considered compromised, as shown in Figure 7.

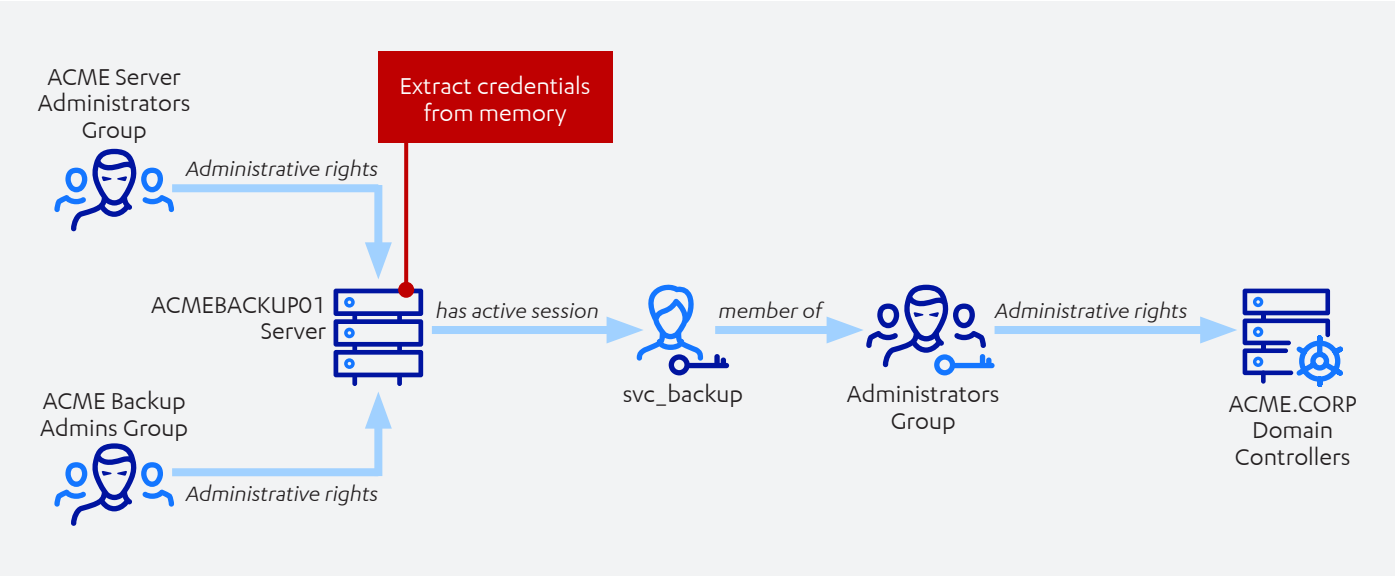


Figure 7: *AD Tier 0 compromise path through enterprise software*

3.2 Tier 0 privileges in the resource forest

In the second example, the organization took the same actions as in the first example, as well as the following additional steps:

1. Cleared out the "Domain Admins", "Administrators", and similar Tier 0 groups in the **acme.corp** corporate forest
2. Service accounts were removed from these groups, and either assigned less privileges, or moved into Tier 0 in the **acme.admin** administration forest

The organization included service accounts and systems in their tier model implementation, and ensured accounts were removed from the groups with Tier 0 privileges in the corporate forest. Only the built-in default Administrator account remained, but this account wasn't in use. The expectation was that Tier 0 in the corporate forest couldn't be compromised, since there were no users in the Tier 0 groups. All Tier 0 accounts and systems were in the administration forest, and were only permitted to log on to Tier 0 resources.

However, some accounts in the corporate forest still possessed Tier 0 equivalent privileges, but weren't present in any of the Tier 0 groups. These accounts were overlooked when Tier 0 was cleared out in the corporate forest. In this example, one of these accounts was the Azure AD Connect service account, called *svc_adconnect*.

Azure AD Connect is used to synchronize accounts between an on-premises domain and Microsoft Azure AD, typically to enable employees to access Office 365 resources using their domain accounts. To synchronize passwords, the service account requires the "Replicating Directory Changes" and "Replicating Directory Changes All" privileges[6], which enables it to obtain the password hashes for all domain accounts. So these privileges are considered Tier 0 equivalent, since they could be used to compromise all accounts in a domain.

AD Compromise Path

The compromise path for this scenario would be very similar to that of the first scenario. The organization didn't realize Azure AD Connect should be a Tier 0 system, so it was installed on a typical server and is now effectively part of Tier 1 in the corporate forest. The server in this example is called ACMEADCONNECT01. The standard set of Tier 1 administration groups were granted administrative privileges to the Azure AD Connect server through Group Policy, as was done with the majority of the other servers in the domain.

Any of the accounts in the Tier 1 server administration groups could be used to compromise the Azure AD Connect server, extract the credentials for the svc_adconnect account, and use it to compromise the corporate forest. This process is illustrated in Figure 8. With a comprehensive tiering implementation, it ideally shouldn't be possible for an attacker to obtain one of these Tier 1 accounts - but this does still constitute a privilege escalation path from Tier 1 to Tier 0. Any flaws in the tiering implementation that allow an attacker to gain access to Tier 1 could be used in conjunction with this issue to compromise the domain.

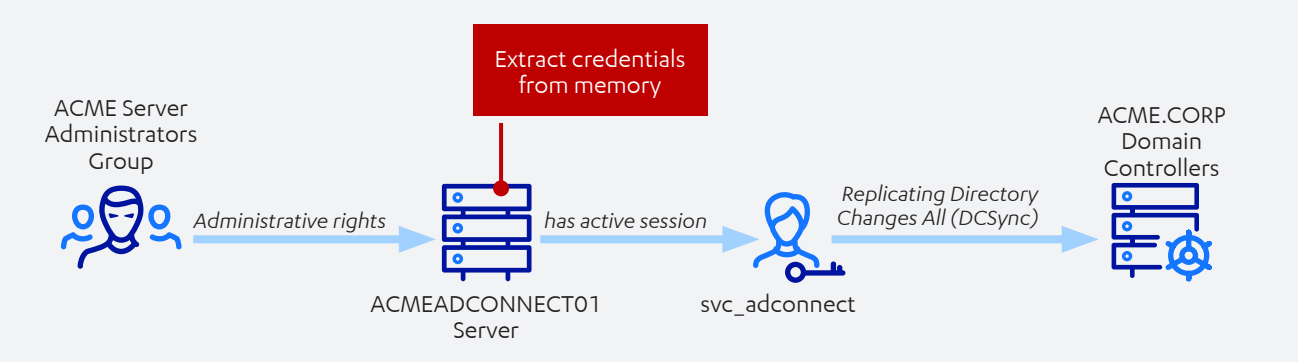


Figure 8: *AD Tier 0 compromise path through Tier 0 permissions left over in the resource forest*

[6] Refer to [6], https://docs.microsoft.com/en-us/azure/active-directory/hybrid/reference-connect-accounts-permissions#ad-ds-connector-account-required-permissions-for-express-settings

3.3 Practical difficulties of implementing tiering

The final example will consist of an environment with the individual misconfigurations from the previous examples, as well as the consequences of some of the practical difficulties of implementing tiering. In this scenario, a separate administration forest was built, and administrative accounts were moved into the administration forest. PAWs were created in the administration forest for all the Tier 0 users. However, the tiering implementation was never fully completed.

Before the tier model had been implemented in the environment, a large number of users' normal domain accounts had been granted administrative rights to one or more hosts in the domain. Many of these accounts' permissions were removed after separate tier accounts had been created in the administration forest for these users. But some of the accounts' permissions were still in place. It was challenging to accurately identify all accounts with administrative rights, since on many hosts the membership of the local Administrators group had been changed manually, rather than through Group Policy.

Another problem was the amount of time and effort required to identify and re-assign privileges from thousands of old accounts to new accounts in the administrative forest without impacting systems, and to determine which privileges were no longer required.

AD Compromise Path

In this final example, a full theoretical AD compromise path will be presented. The starting point for this attack path would also be a compromised workstation after a successful phishing attack. The compromised workstation belongs to the user *Alex*, in the **acme.corp** domain. This user is one of several hundred members of the "Citrix Power Users" group, which has administrative rights to a few Citrix servers in the domain.

STEP 1

The attacker observes an administrator has an active session on one of these Citrix servers, ACMECTX03. The administrator was busy performing maintenance on the server using their Tier 1 administration account, *david_t1*. This account is located in the separate administration forest, **acme.admin**. The attacker then uses Alex's account to deploy a payload to this server, and attempts to extract the *david_t1* account's credentials from memory. However, the attempt fails. The organization had also implemented several other recommended hardening controls on suitable systems. Some of these controls are aimed at preventing credential theft attacks:

- Protected Users group: Credentials for users in this group are not cached in memory, except for Kerberos ticket-granting tickets (TGTs). Additionally, TGTs for users in this group are only valid for four hours, instead of the default ten hours. Other protections also apply, that are beyond the scope of this discussion.
- Credential Guard: This control essentially virtualizes the entire operating system, and ensures the Windows credential provider cannot be accessed from the operating system. With this control, it would no longer be possible to extract credentials directly from memory.

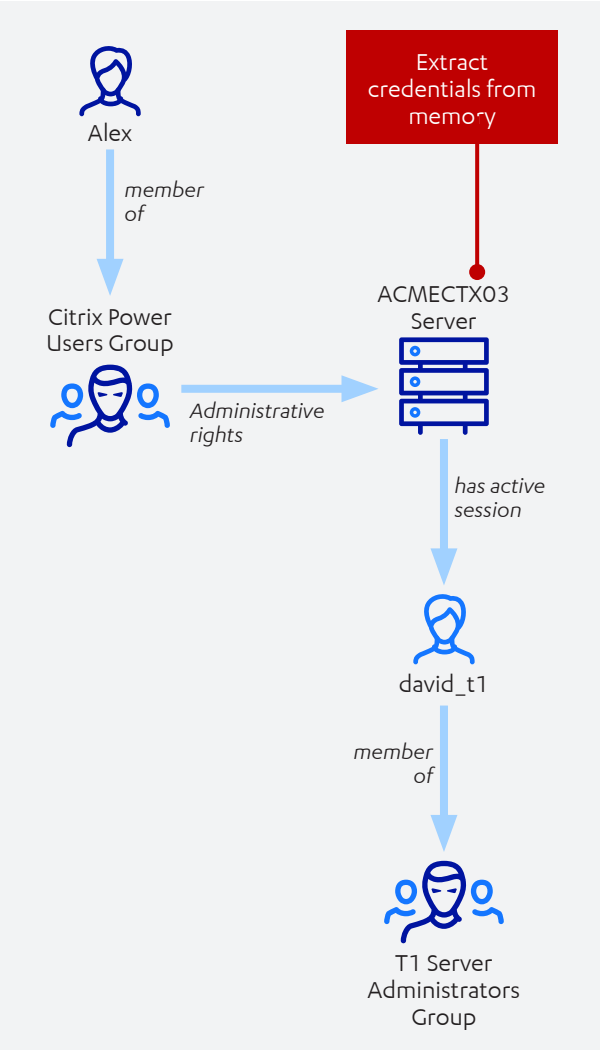


Figure 9: *AD Tier 0 compromise scenario due to an incomplete tiering implementation - step 1*

Due to the presence of these controls, the attacker was unable to extract any credential material for the *david_t1* account. However, with administrative access to this server, the attacker still has a few options for gaining access to the target account. While the administrator is actively logged on to the host, the attacker would still be able to impersonate an access token from one of the administrator's processes, or inject a payload directly into one of these processes. As long as the process remains active, the attacker would be able to impersonate the target user's account on the network.

The attacker decides to perform this attack, and impersonates an access token from one of the *david_t1* account's processes, as shown in Figure 9..

STEP 2

The *david_t1* account is a member of the "T1 Server Administrators" group from the administration forest. According to its role, this group is a member of the local Administrators group on the majority of the servers in the corporate forest. The attacker could therefore use the impersonated access token from this account to compromise any of these servers. The attacker sees that one of the servers this group has administrative rights to is the Azure AD Connect server, ACMEADCONNECT01, from the second example scenario.

Using the impersonated access token for the *david_t1* account, the attacker deploys a payload to the ACMEADCONNECT01 server and extracts the credentials for the *svc_adconnect* account from memory, as shown in Figure 10.

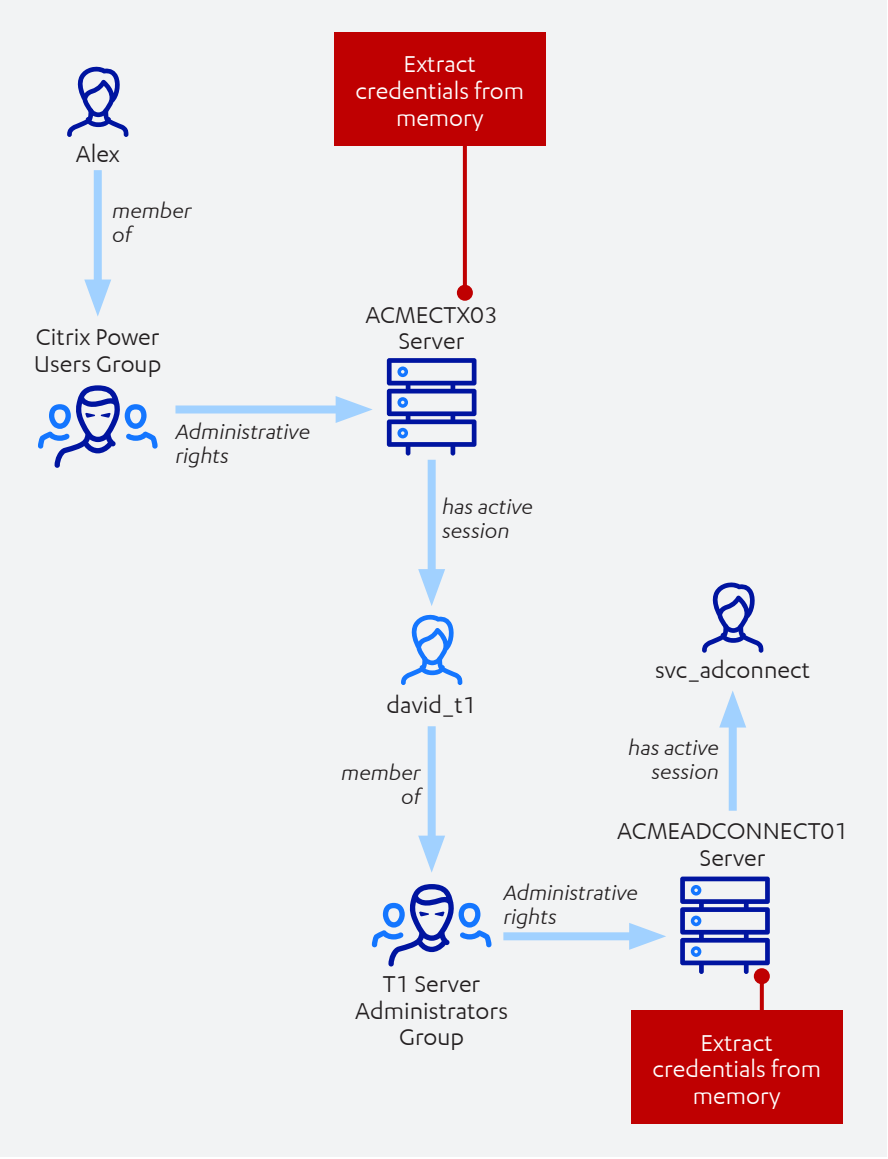


Figure 10: *AD Tier 0 compromise scenario due to an incomplete tiering implementation - step 2*

STEP 3

Finally, the attacker uses the password hash for the *svc_adconnect* account to perform a DCSync attack and retrieve the password hashes for all accounts in the **acme.corp** domain, as shown in Figure 11. The attacker's now able to assume control over virtually any systems or users necessary for them to execute their attack against ACME's key assets.

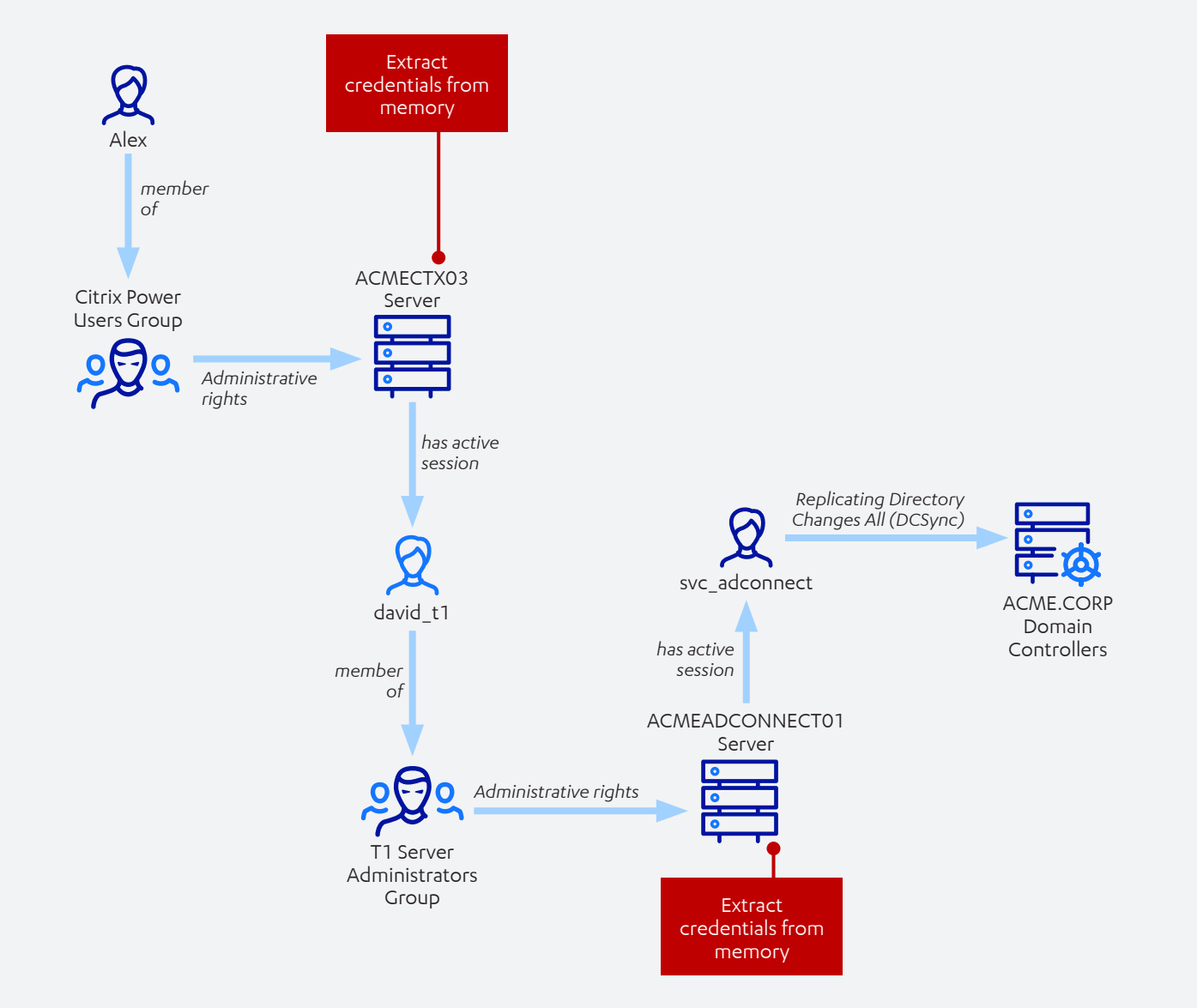


Figure 11: *AD Tier 0 compromise scenario due to an incomplete tiering implementation - step 3*

Despite the presence of a number of additional security controls, including the implementation of a separate administrative Red Forest, it was still possible to compromise the domain in a similar way to the simple example attack presented in Section 1.2.

3.4 Why was this possible?

A number of factors contributed towards these attacks still being possible in our example scenarios, despite the separate administration forests being created and several other controls having been implemented. One fact these examples illustrate is that having administration accounts in a separate forest doesn't inherently provide them with significantly increased levels of protection. Without a strict tier model implementation or additional hardening controls, these administration forest accounts would still be used to log on to systems where low-privileged accounts have administrative rights. They could be compromised using these low-privileged accounts in the same way as before. Essentially the same attack from Section 1.2 would still be possible; the same user's account would be compromised, except it belongs to the administration forest in this case.

There may also be cases where the administrative Red Forest was built and only Tier 0 was created but Tier 1 and 2 was not defined, either in this forest or through a separate PRIV forest. A similar situation would arise, where most of the systems in the domain could be compromised in the same way as before, excluding Tier 0. It would thus take a single misconfiguration or service account with Tier 0 equivalent privileges for Tier 0 to also be compromised in the resource forest.

A possible advantage these implementations do offer is that compromising the administrative forest itself could be significantly more difficult. However, if an attacker's compromised the resource forest, they would most likely already have the necessary level of access required to compromise their target systems and execute their attack, since key business assets will usually reside in the resource forest.

4 REASONS THE IMPLEMENTATION WAS INCOMPLETE AND DIDN'T SOLVE THE PROBLEM

From the examples in the previous section, it's clear these implementations weren't successful in solving the problem meant to be addressed by Red Forest - preventing the compromise of the domain. This section provides further analysis of why organizations may end up with an ineffective implementation of ESAE, and why they don't fully solve the problems they were intended to address.

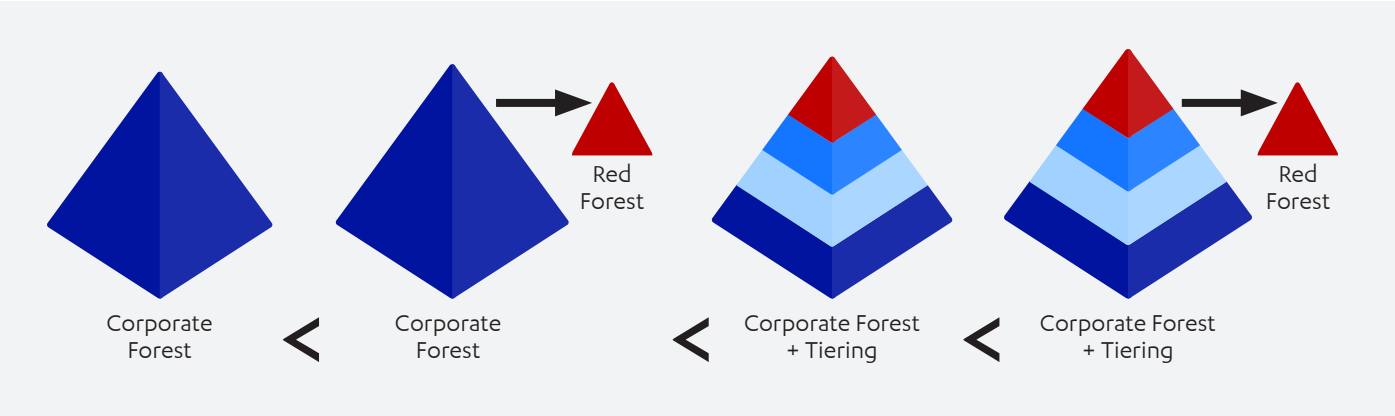
4.1 ESAE is broad and all of its components do not provide the same level of security benefit

As mentioned earlier in this paper, ESAE covers a broad range of security controls, and implementing the full design may not be practical for many organizations. Prioritization is critically important to ensure tangible security benefits are realized within a reasonable timeframe. This prioritization should primarily be driven by the following considerations:

- The aspects of the model that provide the greatest overall improvement to security posture
- Integral parts of the model, without which the security benefits of other aspects would be undermined

For example, the lack of tiering was a significant contributing factor to the attacks described in the previous section. Although a complete implementation of tiering would require a significant amount of time and effort, other aspects of the Red Forest model simply would not be effective without it. Implementing other aspects of the model before implementing tiering, or leaving a tiering implementation incomplete in order to focus on other aspects instead, would most likely end up being relatively ineffective in preventing privilege escalation. **Even a separate administration forest, which does provide limited benefits in isolation, would not significantly improve AD security without a properly implemented tier model.** In comparison, a single corporate forest with a tiering implementation and no separate administration forest would be more effective at preventing attacks. This example is illustrated in Figure 12.

Figure 12: *Comparison between effectiveness of different combinations of tiering and separate administration forests*



Implementing ESAE can seem daunting and some of the finer details may be overlooked when deciding how to prioritize implementation. The model relies on some fundamentally important assumptions and prescriptions for administration processes, required for the model to achieve its intended goal, and aren't simply extra hardening controls. One such example is the use of Restricted Admin mode for Remote Desktop (RDP) when performing remote administration, particularly in Tier 1 and 2. According to Microsoft's documentation, "Securing Privileges Access Reference Material"[7], remote administration, or support for Tier 1 and 2 should be conducted using tools that use network logons (type 3), RDP Restricted Admin mode, or with a local account set through LAPS. The documentation also explicitly states that "Standard RDP may not be used with a domain account" for remote server and workstation support.

The reason standard RDP is so strictly prohibited is because an interactive logon is performed rather than a network logon, in which case credentials are cached on the target host. Although this risk can be mitigated by some controls, such as Credential Guard or the Protected Users group, there are still ways an attacker could impersonate a user. For example, the attacker could impersonate an access token from a target user's process, as explained in Section 2.3.

[7] Refer to [7], https://docs.microsoft.com/en-us/windows-server/identity/securing-privileged-access/securing-privileged-access-reference-material#use-of-approved-support-technology-and-methods

On the other hand, Restricted Admin mode performs a network logon and doesn't cache credentials on the target host. Access tokens for processes are only assigned privileges in the context of that host, so even if an attacker impersonated these access tokens, they couldn't be used to access any other systems over the network. The only way a user could still be compromised is if they entered their credentials on that host in order to access other systems over the network, but this is generally prohibited.

If standard RDP is used to log on to a host, the account used to connect could be impersonated on that host, even if hardening controls are in place. With Restricted Admin mode, impersonation would not be possible.

Restricted Admin mode is one of the most powerful controls available for preventing credential theft attacks, which is why it's required for administration processes according to ESAE.

Despite this, Restricted Admin mode has rarely been observed in client networks assessed by F-Secure Consulting. This could be due to concerns around Restricted Admin mode enabling Pass-the-Hash attacks for RDP. However, the situation isn't quite that simple, and the benefits still outweigh the potential risks overall[8]. Regardless of the Pass-the-Hash considerations, Restricted Admin mode for RDP is a requirement when using tiered accounts to administer machines in lower tiers using RDP, to prevent privilege escalation opportunities.

Enforcement of a control like Restricted Admin mode should be considered one of the main priorities for an ESAE implementation, since its widespread use can cripple credential theft attacks and opportunities for privilege escalation. Instead, it easily becomes lost in the details of implementing the larger model. If the right aspects of the Red Forest model aren't prioritized, the eventual implementation may fall well short of achieving its intended goal - securing AD. This can leave organizations with a false sense of security a few years down the line after starting with a Red Forest implementation.

4.2 Some controls can end up being almost completely ineffective if they're not implemented correctly

A challenge with preventing many of the attack techniques used in the context of AD is they rely on using functionality in unexpected or unintended ways. If any given control isn't implemented carefully, the attacker may simply perform a different, unexpected action and bypass the control entirely. To avoid this, the following should be considered when implementing the controls in ESAE:

- Follow the recommended approach as closely as possible.
- Where deviations or changes are necessary, consider the implications carefully.
- **Understand the attack a control is meant to prevent - this would enable you to determine whether a control would still work as intended if any changes or deviations are made.**

A simple example to illustrate this point is enabling the Windows Firewall on a workstation to prevent lateral movement. The correct implementation would be to block all inbound access, except for access to specific ports from specific systems. Examples of these exceptions may include allowing inbound access to SMB from SCCM, or allowing access to RDP from the desktop support team's IP addresses.

An incorrect implementation would be to only block inbound access to RDP. Blocking access to RDP might seem like it would prevent an attacker from accessing the workstation but, in reality, most lateral movement techniques make use of services such as SMB and RPC. Without an understanding of the attacks this control is intended to prevent, it's likely mistakes would be made during implementation. This is especially true when adjustments need to be made, such as when systems X and Y cannot perform their intended functions anymore because of access to SMB being blocked. Rules would need to be created to grant only systems X and Y access to SMB, instead of simply allowing access to SMB from any source, since that would completely invalidate the control.

[8] Refer to [8], https://labs.f-secure.com/blog/undisable/

5 RECOMMENDED PRIORITIZATION OF CRITICAL ESAE COMPONENTS

The point of highlighting these problems and pitfalls is ultimately to determine what would be required for a Red Forest implementation to be successful.

As the example attacks illustrate, **a complete tiering implementation is absolutely integral to the Red Forest model**.

Without it, many of the other components wouldn't be nearly as effective as they should be. There are also a few specific controls that could significantly improve a domain or forest's security posture, without requiring a lot of effort to implement or significantly disrupting operations. These controls should ideally also be prioritized in a Red Forest implementation.

To adequately protect an AD environment against attacks, the following components should be assigned a high priority as part of a Red Forest implementation:

- Implement a complete and comprehensive tier model
- Implement privileged access workstations (PAWs) for Tier 0
- Use Restricted Admin mode for RDP on all suitable machines
- Address service accounts with excessive privileges and weak passwords
- Identify and remediate misconfigurations that could be used to compromise privileged accounts
- Enable the Windows Firewall on workstations and restrict access to remote administration services (e.g. SMB and RDP)
- Secure local account management
- Implement a PAM solution

This list is by no means a comprehensive set of controls for securing AD, but it is a good starting point. Although some of these components would likely be complex and time-consuming to implement, a Red Forest implementation would be ineffective without them. If an organization does implement these components successfully, it would make their AD environment significantly harder to compromise.

5.1 Implement a complete and comprehensive tier model

The tier model is arguably the most important aspect of Red Forest - without the privilege separation provided by a tier model, privilege escalation opportunities would always exist and many other controls could be circumvented. It's important that the tier model is fully completed, since even small gaps in the implementation could still be exploited by an attacker. Many different factors need to be considered when separating users, systems, and privileges into tiers, including:

- Local administrator privileges
- ACLs granting access over objects in AD
- Privileges to edit GPOs
- Access to enterprise systems that indirectly provide privileges over machines, for example:
  - SCCM - pushing arbitrary packages
  - Hypervisor management consoles for virtualized servers - accessing disk and memory snapshots
  - Backup solutions - accessing disk images and retrieving local account password hashes
- Access to enterprise systems that indirectly provide privileges over user accounts and other AD objects, for example:
  - Helpdesk solutions - performing password resets on privileged accounts
  - AD management software - assuming control over privileged AD objects

If any of these aspects are overlooked, it could lead to a break between tiers. For example, SCCM, backup solutions, and virtualization management consoles are often managed by server administration teams (Tier 1), but also used for domain controllers (Tier 0). These Tier 1 administrator accounts would be able to use these systems to gain access to domain controllers and compromise Tier 0.

Correctly integrating service accounts into a tier model can be particularly challenging, and may even be dependent on software vendors. These considerations are discussed in Section 4.4.

In an environment where users didn't previously have separate administration accounts, a lot of work would be required to:

- Determine which privileges each user requires
- Create the necessary tier accounts for them
- Assign the required privileges to the tier accounts
- Remove all the privileges from their normal accounts

This would become a manual process to some extent, since users may have been granted privileges previously they don't actually require. Also, testing would need to be done to ensure each user's new tier accounts work correctly for all the systems they need to access.

This process could be followed in conjunction with building separate Admin and PRIV forests, as long as the focus is on completing the tiering implementation, rather than simply building separate forests. Once defined, the tier accounts would then be created directly in the administration forests, and any systems that need to be moved into different tiers would then also be moved into those forests.

Alternatively, tiering could be implemented in the current forest first, in which case the majority of the benefit would already be realized. If the organization then decides to implement separate administration forests afterwards, it would be a relatively simple process of moving all the tier accounts, groups, and systems into the administration forests and replicating their privileges. The largest portion of work is defining the tiers, creating all the tier accounts and systems, determining which privileges they require, and stripping privileges from normal accounts, which would have already been done by that point.

5.2 Implement privileged access workstations for Tier 0

Another critical part of the Red Forest model is the concept of a privileged access workstation. The intention behind PAWs is to prevent a break in the tier model that would be caused by desktop administrator accounts (Tier 2) being able to compromise Tier 1 and Tier 0 administrators, by assuming control over their workstations. Additionally, PAWs would limit the impact of a successful phishing attack against an administrator, by ensuring the account and environment they use for browsing the internet, viewing emails, and performing general business tasks is separated from their privileged accounts.

A PAW can be implemented in a few different ways, including:

- A separate physical workstation used for administration activities.
- A single workstation configured as the PAW, with a virtual machine used for general business tasks.
- A single workstation configured as the PAW, where the user connects to a VDI or Remote Desktop server to perform general business tasks.

In each of these cases, the configuration needs to ensure the PAW cannot be accessed from the workstation or virtual machine used for general business tasks, and that the privileged account cannot be used to log on to the business workstation or virtual machine. The PAWs belonging to administrators in a particular tier should only be accessible to specific users within that tier, and no users or systems in a lower tier should have any form of administrative access or control over the PAWs.

Some challenges may arise in the practical implementation of PAWs, such as software and update management, and backups. Since the PAWs aren't permitted to access the internet, an internal update server or SCCM instance would be required to distribute updates to the PAWs. However, if the current SCCM instance is not in Tier 0, a separate instance would need to be created specifically for the PAWs and possibly other Tier 0 systems. Without this, Tier 1 accounts would be able to use SCCM to compromise the PAWs by pushing arbitrary packages to them.

According to the Red Forest model, all administrators across all tiers would need to make use of PAWs. Although this would be the ideal case, it could be a significant undertaking for a large organization with hundreds or thousands of administrators. In some cases, these administrators' existing workstations may not have the necessary specifications to run virtual machines without significantly degrading performance. So they would first need to be provided with new workstations before a PAW setup would be feasible. For this reason, the highest priority would be to first provide Tier 0 users with PAWs, which should be a relatively small number. Enabling the Windows Firewall on the rest of the administrators' workstations, as discussed in Section 4.6, would then act as a mitigating control until the organization's ready to implement PAWs for Tier 1 and 2 as well.

A full description of privileged access workstations is available in Microsoft's documentation[9]. This includes detailed configuration instructions and PowerShell scripts to automate some parts of the process.

5.3 Use Restricted Admin mode for RDP on all suitable machines

Restricted Admin mode for RDP was discussed in some detail in Section 3.1. It's an excellent control for defending against credential theft attacks. And, according to the Red Forest model, is the only way in which Remote Desktop is allowed to be used for Tier 1 and 2 - using standard RDP with a domain account is forbidden in these cases. Restricted Admin mode was introduced with Windows Server 2012 R2, but has since been backported to earlier versions as well. So it should be possible to enable it on all suitable servers in a network.

There are cases where Restricted Admin mode wouldn't be appropriate or effective.
For example, any servers from where users would connect to additional servers or systems and enter their domain credentials, such as a jump box. In some cases these credentials would then be cached on the server, and an attacker would also be able to keylog the users while they enter them. Enabling Restricted Admin mode on such a server wouldn't provide much benefit, since it would still be possible for the users' credentials to be compromised.

[9] Refer to [4], https://docs.microsoft.com/en-us/windows-server/identity/securing-privileged-access/privileged-access-workstations

5.4 Address service accounts with excessive privileges and weak passwords

When implementing a tier model, service accounts shouldn't be overlooked. Highly privileged service accounts could easily result in a break in the tier model, even if they're not direct members of groups like Domain Admins. If a service account is a member of any Tier 0 equivalent groups[10], possesses Tier 0 equivalent privileges (e.g. "Replicating Directory Changes All" privileges), or possesses any privileges that grant it control over Tier 0 entities, it could result in a privilege escalation opportunity if it's used on systems outside of Tier 0.

Systems installed on Tier 1 infrastructure that have privileged service accounts would need to be investigated. If the service account is found to be highly privileged, one of the following actions would need to be taken:

- If the system doesn't require access to or control over any Tier 0 resources, its privileges could be removed, and a more limited set of privileges could be assigned instead.
- If the system requires access to or control over any Tier 0 resources, the system may need to be moved into Tier 0 itself. Alternatively, a separate dedicated Tier 0 instance of the system could be installed, and the original instance could have its privileges reduced and used for the rest of the environment.

Another common problem with service accounts, particularly very old ones, is the use of weak passwords.

A simple password guessing attack could be a very effective method to escalate privileges, even in a tiered environment.

PowerShell AD modules can be used to retrieve the time when passwords were last changed for all accounts in AD, as well as which accounts are configured with passwords that never expire. This list could then be sorted according to the longest time since the last password change. If an account's password hadn't been changed in several years or more, it's likely the account may have a weak password, and wouldn't conform to stricter password policies that could have been implemented since then. This often happens with legacy systems that don't have a simple process for changing their service accounts' passwords, or systems where the passwords were hard-coded.

Solutions are available to simplify the management of service account credentials. In AD itself, service accounts could be replaced with group Managed Service Accounts (gMSA)[11]. The passwords for these accounts are automatically managed by Windows. Some PAM solutions can also be used to manage service account passwords centrally.

There are many factors to consider when addressing these issues and attempting to fully integrate service accounts into a tier model:

- Understanding how a software solution uses its service account credentials (e.g. interactive vs. network logons).
- How many machines a service account logs on to, and on which machines its credentials are cached.
- How reliant the organization has become on a particular software solution, if it turns out to be poorly architected and exposes its service account credentials to a large portion of the network.
- If changes need to be made to the way in which a solution uses its service account credentials, it may require the involvement of the software vendor.

[10] Refer to [7], https://docs.microsoft.com/en-us/windows-server/identity/securing-privileged-access/securing-privileged-access-reference-material#tier-0-equivalency

[11] Refer to [9], https://docs.microsoft.com/en-us/windows-server/security/group-managed-service-accounts/group-managed-service-accounts-overview

5.5 Identify and remediate misconfigurations that could be used to compromise privileged accounts

There are various misconfigurations and vulnerabilities that could be used to directly compromise privileged accounts and systems in AD. Some of these misconfigurations may exist in other systems that are closely integrated with AD, and not in AD itself. A few examples are:

- **Using privileged account credentials in the McAfee Agent for accessing update repositories on the ePolicy Orchestrator (ePO) servers:** The McAfee enterprise AV agent can be configured to retrieve updates from UNC shares on the ePO servers. In order to access these shares, the agents need to be configured with an account that has the necessary rights to access those shares. These credentials are stored in the agents where it's possible to retrieve them[12]. A commonly observed misconfiguration is to use a highly privileged account for this purpose. The agent would likely be installed on all or most of the machines in a network, including low-privileged users' workstations, and the credentials could be retrieved from any of these machines. The recommended configuration is to use a low-privileged account that only has rights to access the shares[13].
- **Weak credentials or misconfigurations in AD management software:** Enterprise AD management solutions typically have highly privileged access in AD, and should ideally be protected in a similar way to domain controllers. Misconfigurations in these solutions, such as neglecting to disable default administration accounts, could lead directly to a domain compromise.
- **Not enforcing SMB signing:** Researchers from SpectreOps discovered the Windows Print System Remote Protocol (MS-RPRN) could be used to coerce a Windows host to authenticate to other hosts, which has several notable implications for security[14]. This authentication attempt can be directed to an attacker-controlled machine, where it could be modified and relayed to a different host if SMB signing isn't enforced. If a host's machine account is a member of the local Administrators group on any other hosts, this technique could be used to compromise those hosts. This often includes highly privileged systems; for example, the SCCM installation documentation states that the SCCM servers' machine accounts require administrative access to the SCCM database server. Any low-privileged user can issue the required requests to MS-RPRN in order to exploit this issue, allowing them to compromise the SCCM database server. SMB signing is enabled and supported by default, but not required. Windows hosts need to be configured to require signing to prevent this type of attack.
- **Remote Code Execution (RCE) vulnerabilities on critical servers:** Vulnerability management can be a critical component that ensures the overall integrity of the AD environment. If servers are vulnerable to RCE vulnerabilities, and these servers store credentials for privileged Tier 1 administrators in memory (such as in the absence of Restricted Admin mode for RDP), the vulnerability itself is the cause of a privilege escalation opportunity.

These misconfigurations don't necessarily fit neatly into any specific part of the Red Forest model, and in some cases involve systems outside of Active Directory. However, they could be exploited by an attacker to gain access to privileged accounts, or in some cases directly compromise AD itself. This would be considered a breach in the tier model. Even with a correctly implemented Red Forest, AD could still be compromised if these types of issues aren't identified and resolved.

The difficult part of resolving these types of misconfigurations is identifying them in the first place, since almost any system integrated with AD could be affected. An effective solution might be to conduct internal penetration tests, preferably with a focus on compromising Active Directory. Such an exercise should identify some of these types of issues. This should likely be complemented with vulnerability scanning which, in combination with penetration testing, should feed into a vulnerability management program that prioritizes addressing these misconfigurations and vulnerabilities as a key part of remediation.

---

[12] The relevant scripts and descriptions are available in [10] https://gist.github.com/toufik-airane/919d168c35d810861eb7b2f68723afbb and [11] https://github.com/funoverip/mcafee-sitelist-pwd-decryption

[13] https://kc.mcafee.com/corporate/index?page=content&id=KB70999

[14] Refer to [12], https://www.slideshare.net/harmj0y/derbycon-the-unintended-risks-of-trusting-active-directory

5.6 Enable the Windows Firewall on workstations and restrict access to remote administration services

In many environments, the Windows Firewall (or similar host-based firewall solution) is disabled on domain-joined Windows hosts. **If configured correctly, a host-based firewall can be an extremely effective control for preventing lateral movement.** Even if an attacker managed to obtain administrative credentials for a host, they wouldn't be able to log on or deploy a payload to that host if they cannot access any network services on it. The Windows Firewall could be disabled by editing a GPO linked to the target host; however, only highly privileged accounts typically have write access to GPOs, so effectively the attacker would only be able to do so after they have already compromised the domain.

If the Windows Firewall is enabled, it should be configured to only allow access to specific services from specific sources where it's absolutely required. On servers this can be difficult to achieve in practice, as there are often many different connections between servers, and many users that need to access them from across the entire network. On the other hand, there are very few legitimate reasons for inbound connections to user workstations. Therefore enabling the Windows Firewall specifically on workstations could be an effective quick-win for mitigating lateral movement in a network. Some examples of exceptions that would need to be created are as follows:

- **Remote Desktop Support:** If the organization provides remote desktop support to employees, an exception would need to be added for desktop support teams to connect to RDP (and possibly SMB and RPC) on the workstations. The desktop support team would then need to be assigned static IP addresses or operate from a specific network range, in order for firewall rules to be created. Alternatively, they could make use of a secure Tier 2 jump box for this purpose.
- **SCCM:** If SCCM is configured to push packages to endpoints, an exception would need to be created to allow SCCM to connect to SMB on workstations.
- **Other Enterprise Software:** Any other enterprise software solutions that need to push packages or updates to endpoint agents, as opposed to solutions where the endpoint agents pull updates from the servers, would also require firewall rules to allow inbound access to workstations.

The firewall should be configured to block all inbound access by default, with exceptions only for specific IP addresses as in the examples above. The following ports are often used by attackers for lateral movement and organizations must ensure they can only be accessed from specific IP addresses:

- **[tcp/135]** RPC
- **[tcp/139]** SMB over NETBIOS
- **[tcp/445]** SMB over TCP
- **[tcp/3389]** Remote Desktop
- **[tcp/5985]** Windows Remote Management (HTTP)
- **[tcp/5986]** Windows Remote Management (HTTPS)

This control would be particularly effective in the early stages of an attack, while the attacker is still attempting to escalate privileges after an initial compromise. At this stage, they're unlikely to have the means to disable or circumvent the firewalls, and wouldn't be able to gain access to any machines where the firewall's enabled.

Windows Firewall is used as the specific example here, since it's present on Windows hosts by default and can be configured through Group Policy. However the general principle of using a host-based firewall to block the ports that could be used for lateral movement can be implemented with any robust enterprise firewall solution that's installed on all endpoints.

5.7 Secure local account management

Using a solution for local account management, such as Microsoft Local Administrator Password Solution (LAPS), is highly recommended. Some PAM solutions may also provide functionality to manage local accounts. The purpose of such a solution would be to ensure:

- Strong passwords are set for local administrator accounts on Windows hosts
- Passwords aren't shared between different hosts
- Administrators are still able to access and use these passwords in a reasonable way when they need them

Without such a solution, the passwords for local administration accounts are often shared between many different hosts, for example as part of a standard build. Even if this password is very strong, an attacker could retrieve the account's password hash if they managed to compromise one of these hosts. They could then use it to compromise all other hosts where the same shared password's used.

5.8 Implement a PAM solution

Microsoft suggests the use of their privileged access management solution, Microsoft Identity Manager (MIM), as part of the Red Forest model. Focus is placed on using the advanced features MIM provides, such as Just Enough Administration (JEA) and Just In Time Administration (JIT), to provide further protection to privileged accounts. A suitable third-party solution could, of course, also be used.

A PAM solution would be very beneficial for administrators when an AD tier model is implemented. Some administrators would have several different accounts to manage, and setting a strong, unique password for each of them can be burdensome. The PAM solution would provide them with a way to manage the passwords for their accounts. It would also provide additional benefits, such as being able to manage certain service account passwords centrally.

Although a PAM solution can provide many benefits, organizations should be careful with how the solution's implemented and ensure it's secured appropriately. Since it would be used to manage highly privileged accounts, it presents an extremely attractive target to attackers. These solutions should either be deployed in Tier 0, or a separate instance should be used for each tier.

A PAM solution deployed in Tier 1, but used to manage Tier 0 accounts, would be a convenient privilege escalation opportunity for attackers. In this case, it would also end up being detrimental to the environment's security posture rather than improving it.

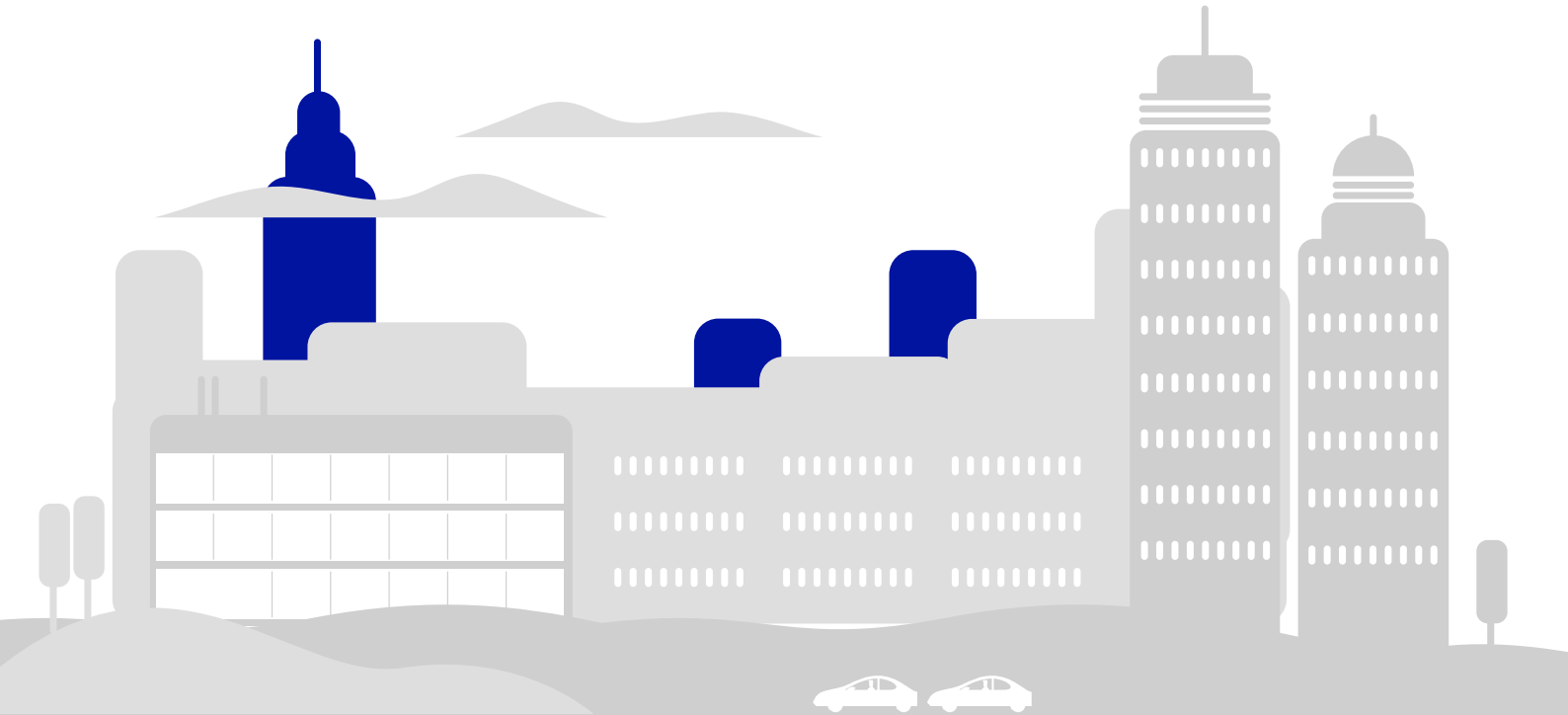
Securing a PAM solution adequately would likely require tiering to be in place, or at least for Tier 0 to be properly defined. The tier model would ensure the PAM solution can be protected in Tier 0, while the PAM solution would strengthen the tier model by providing additional controls and simplifying account management.

6 CONCLUSION

Microsoft's ESAE framework, also known as Red Forest, provides organizations with a comprehensive set of guidelines for securing their AD environment. However, it's often not feasible to implement the full framework or follow the guidelines exactly. In practice, organizations would more likely end up implementing only certain aspects of the framework. Deciding which aspects are most important and how to assign priorities during implementation is critical to ensuring a Red Forest implementation achieves its stated goal. This requires an overall understanding of the attacks that the framework is intended to defend against, which attacks individual controls are applicable to, and how these controls work.

A few examples of incomplete Red Forest implementations and the attacks that could be launched against them were presented. Based on these examples and F-Secure Consulting's experience assessing AD environments with various levels of security maturity, certain aspects of the Red Forest model were identified as being integral to its success. The most notable is the administrative tier model, which the ESAE framework is based on. Without a complete and comprehensive tier model implementation, the rest of the framework would be undermined to a large extent. As a result, it would most likely not provide the expected level of improvement in security.

A list of components were provided in Section 4 that should be assigned a high priority in a Red Forest implementation, or any AD security improvement plan. Some of these components are required for the overall implementation to be effective, such as the administrative tier model, while other often-overlooked components are highly effective without being overly complex to implement. If these components are implemented correctly, they would significantly improve the security posture of an organization's AD environment, increasing its overall resilience towards targeted attacks. If desired, the organization would then be primed to commence with a full Red Forest implementation to secure the environment even further.



7 REFERENCES

[1] S. Metcalf, "Cracking Kerberos TGS Tickets Using Kerberoast – Exploiting Kerberos to Compromise the Active Directory Domain," [Online]. Available: https://adsecurity.org/?p=2293.

[2] W. Schroeder, "Roasting AS-REPs," [Online]. Available: https://www.harmj0y.net/blog/activedirectory/roasting-as-reps/.

[3] S. Metcalf, "Mimikatz DCSync Usage, Exploitation, and Detection," [Online]. Available: https://adsecurity.org/?p=1729.

[4] Microsoft, "Privileged Access Workstations," [Online]. Available: https://docs.microsoft.com/en-us/windows-server/identity/securing-privileged-access/privileged-access-workstations.

[5] Microsoft, "Privileged Access Management for Active Directory Domain Services," [Online]. Available: https://docs.microsoft.com/en-us/microsoft-identity-manager/pam/privileged-identity-management-for-active-directory-domain-services.

[6] Microsoft, "Azure AD Connect: Accounts and permissions," [Online]. Available: https://docs.microsoft.com/en-us/azure/active-directory/hybrid/reference-connect-accounts-permissions#ad-ds-connector-account-required-permissions-for-express-settings.

[7] Microsoft, "Active Directory administrative tier model," [Online]. Available: https://docs.microsoft.com/en-us/windows-server/identity/securing-privileged-access/securing-privileged-access-reference-material#use-of-approved-support-technology-and-methods.

[8] D. Yates, "Undisable Restricted Admin," [Online]. Available: https://labs.f-secure.com/blog/undisable/.

[9] Microsoft, "Group Managed Service Accounts Overview," [Online]. Available: https://docs.microsoft.com/en-us/windows-server/security/group-managed-service-accounts/group-managed-service-accounts-overview.

[10] T. Airane, "McAfee privileged SiteList.xml leads to Active Directory domain privilege escalation," [Online]. Available: https://gist.github.com/toufik-airane/919d168c35d810861eb7b2f68723afbb.

[11] J. Nokin, "Password decryption tool for the McAfee SiteList.xml file," [Online]. Available: https://github.com/funoverip/mcafee-sitelist-pwd-decryption.

[12] McAfee, "Recommendations for download credentials when using UNC shares as software repositories in ePolicy Orchestrator," [Online]. Available: https://kc.mcafee.com/corporate/index?page=content&id=KB70999.

[13] SpectreOps, "Derbycon - The Unintended Risks of Trusting Active Directory," [Online]. Available: https://www.slideshare.net/harmj0y/derbycon-the-unintended-risks-of-trusting-active-directory.


F-Secure.