Restricting SMB-based lateral movement in a Windows environment

medium.com/palantir/restricting-smb-based-lateral-movement-in-a-windows-environment-ed033b888721

Palantir July 8, 2020

Palantir
Follow
Jul 8, 2020

.

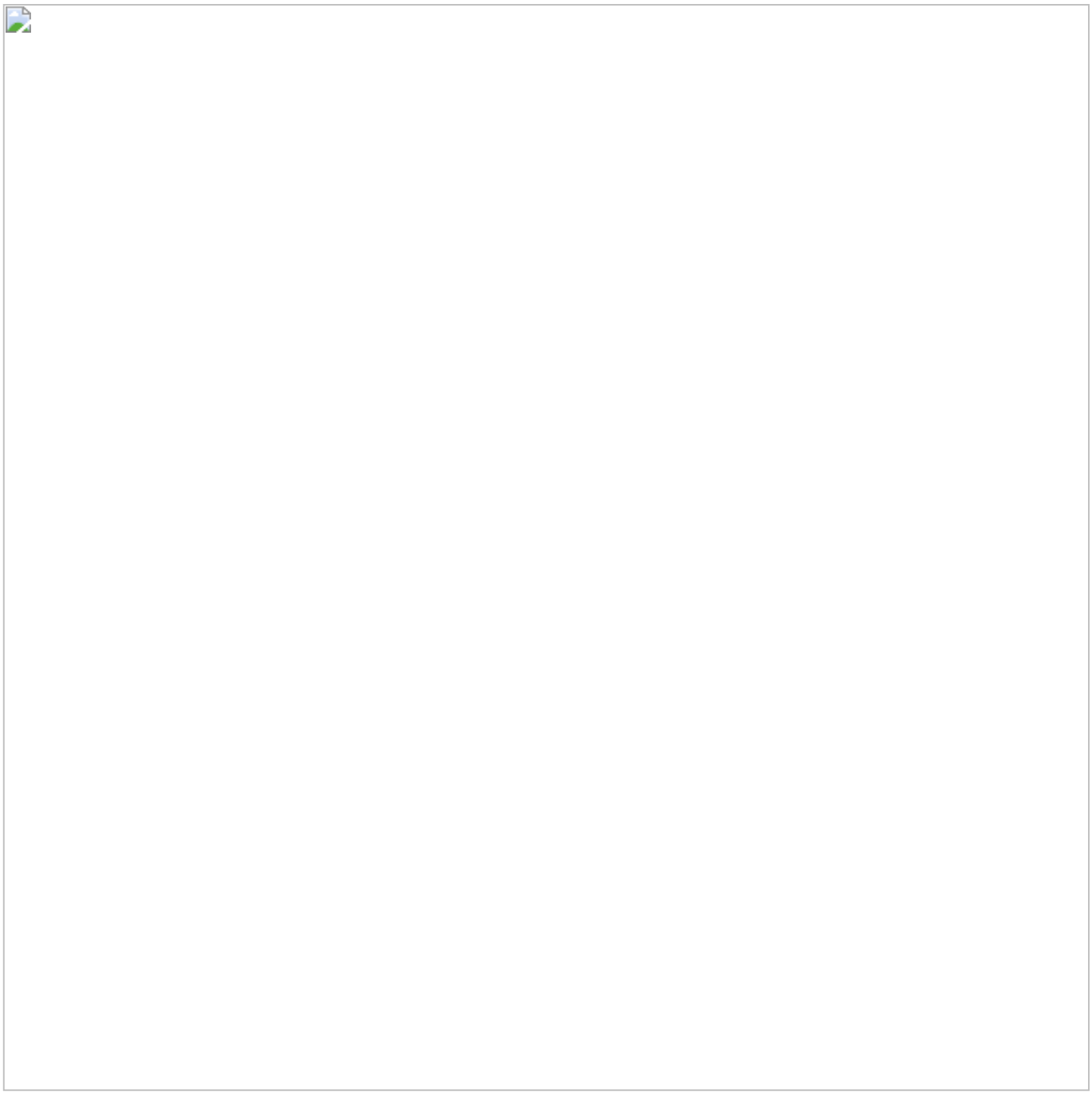
19 min read

When Palantir entered into a technical collaboration partnership with SpecterOps in 2018, one of our key initiatives was the advancement of defensive capabilities against the latest Windows security tradecraft. To that end, the Adversary Simulation team at SpecterOps perform regular red team engagements inside the various Palantir networks and leverage their latest tools and techniques to provide a continual feedback loop for defensive security improvement.

Lateral movement via Windows Server Message Block (SMB) is consistently one of the most effective techniques used by adversaries. In our engagements with the SpecterOps team, this mechanism is consistently targeted for abuse.

Even in networks where significant efforts have been made to eliminate unnecessary SMB exposure, there are usually a small number of servers with a business-critical need to serve files to legitimate clients. In most organizations, the list will include Domain Controllers (the SYSVOL share), File Servers, and PowerShell logging servers, at the very least.

This blog post aims to consolidate the defensive information we've compiled in our efforts to restrict SMB-based lateral movement, after many iterations against the SpecterOps Adversary Simulation Team.



Why is SMB-based lateral movement effective?

At a high level, Server Message Block (SMB) is a network communication protocol that can provide shared access to services on a network. SMB is well-known for file services and for printers, but it's much more versatile than that. **It can also provide an authenticated inter-process communication mechanism between nodes.**

Ryan Hausknecht (SpecterOps) published an excellent blog that goes into detail on Offensive Lateral Movement including details about SMB here.

For the purpose of this defensive blog post, we can oversimplify Ryan's post and say that when an adversary has both:

- Network access to the SMB service (TCP port 139 or 445) and
- Valid credentials

there's high probability that they'll be able to gain code execution on the remote host and expand their attack surface in the environment.

Bypassing MFA

The risk created by SMB is especially important in mature environments where multi-factor authentication is required for administrative access to servers.

SMB can provide a convenient MFA bypass for adversaries, handing them a foothold that will allow for remote code execution without any additional authentication factor. Depending on the environment, SMB may also provide adversaries the ability to disable security controls (including MFA) and improve their position in the network.

Example:

> In many environments which implement a 3rd party MFA provider, an attacker can remove the MFA restriction with their SMB-based shell. They achieve this by enabling 'Restricted Admin Mode'. This seems counterintuitive, but works for many MFA clients because restricted admin mode changes the supported Windows logon types and takes the MFA provider out of the picture. The setting is controlled via a single registry key: "HKLM\System\CurrentControlSet\Control\Lsa\DisableRestrictedAdmin" and setting it to 0 is all that is required in many cases. This single change gives the adversary the option of single-factor RDP access to machines that would have otherwise been protected with an MFA prompt.

Summary of Controls

We understand that readers will be operating in networks with various levels of operational maturity, and not all of the recommendations will be practical for every environment. Use this summary to skip forward to the sections/steps that are relevant to your situation.

The controls that have been effective for Palantir are as follows:

- Implementing a simple three-tiered administration model (Workstations, General Servers, Authentication Servers)
- Denying logon for security principals in the wrong tier
- Denying SMB communication between workstations
- Denying SMB communication from workstations to servers
- Prioritizing Operating System upgrades for 'high risk' servers
- Deploying Windows Defender Code Integrity rules

- Deploying Windows Defender Attack Surface Reduction rules

We also believe it is worth calling out the options that we haven't yet been able to derive value from:

- Denying network logon for security principals likely to be used for adversary lateral movement*
- Denying 'log on as a service' for administrative accounts
- Windows Token filtering policies
- Improved Service Control Manager ACLs
- WMI access restrictions
- Advanced Windows Firewall configurations for all SMB traffic — IPSEC (null encryption)**
- Network-based tiering restrictions on a per service level
- Windows Firewall built in Named Pipe rules

> * We begin many of our red team engagements with the assumption that some low-level, standard user credentials have been stolen from an endpoint and the operator has remote code execution on that device.
>
> ** We make heavy use of Windows Firewall and IPSec with null encryption for other use cases.

Effective Mitigations

[1]: Implementing a simple tiered administration model (Workstations, General Servers, Authentication Servers)

As best as we can, we follow Microsoft guidance in our approach to setting up a tiering model for each of our environments. Microsoft have excellent documentation on the topic, including an implementation guide here.

We take a simple approach and divide our environments into three asset types:

- Tier 0 — Domain Controllers, ADFS servers and others that have direct ability to influence authentication.
- Tier 1 — All other servers
- Tier 2 — Workstations

Then, for each of the tier types we restrict the security principals that can log on. Example:

- Tier 0 — 'username-t0' + require MFA.
- Tier 1 — 'username-t1' + require MFA.
- Tier 2 — standard username and MFA depending on the service and scenario.

We restrict all administrative ports to bastion hosts for each tier.

Even if an adversary is in possession of stolen administrative credentials (tier 1 or tier 0), the administrative ports such as RDP and WinRM are not available from lower tier machines.

Administrative ports are only open to traffic from bastion hosts, which are protected with a 3rd party MFA provider. We achieve this with Windows host-based firewall and back it with network firewall rules.

> Palantir CISO Dane Stuckey's 'Endpoint Isolation with Windows Firewall' post provides detail for configuring Windows firewall to suit this model.

In a perfect world, this level of separation would mean that even if an adversary stole tier 1 or tier 0 credentials from a workstation, they would not be able to use them. They'd require MFA to position themselves on a host with access to administrative ports on their targets. Unfortunately, this is where SMB steps in and provides options for abuse.

This first foundational step provides the following significant improvements:

- Attackers need to be on a bastion host to access administrative ports for servers
- Attackers need specific credentials to move between tiers and users do not use those credentials on workstations.

[2]: Denying logon for security principals in the wrong tier

Users are prevented from logging in to a workstation with a tier 1 (server administration) or 0 (domain administration) accounts.

Without extra effort on the side of the adversary, processes can't execute in the context of a tier 0 or 1 security principal on lower tier devices.

We use the **"**Allow log on locally" Group policy for this, linked to the top Organizational Unit for each service tier. The policy denies logon to the two groups containing principals for each of the other two tiers.

This simple policy prevents attackers from logging on to our workstations with server credentials. Interestingly, it also provides protection against remote code execution via SMB if an adversary was to execute:

> runas /user:contoso\bob-t1 'psexec.exe \\tier1server.contoso.com cmd.exe'

Unfortunately, there's a gap in this control as it applies to SMB-based lateral movement. If an attacker is able to leverage a 'network only' logon type, this control will fail. Legitimate tools like psexec have a '/netonly' parameter, as do adversary frameworks.

The improvements:

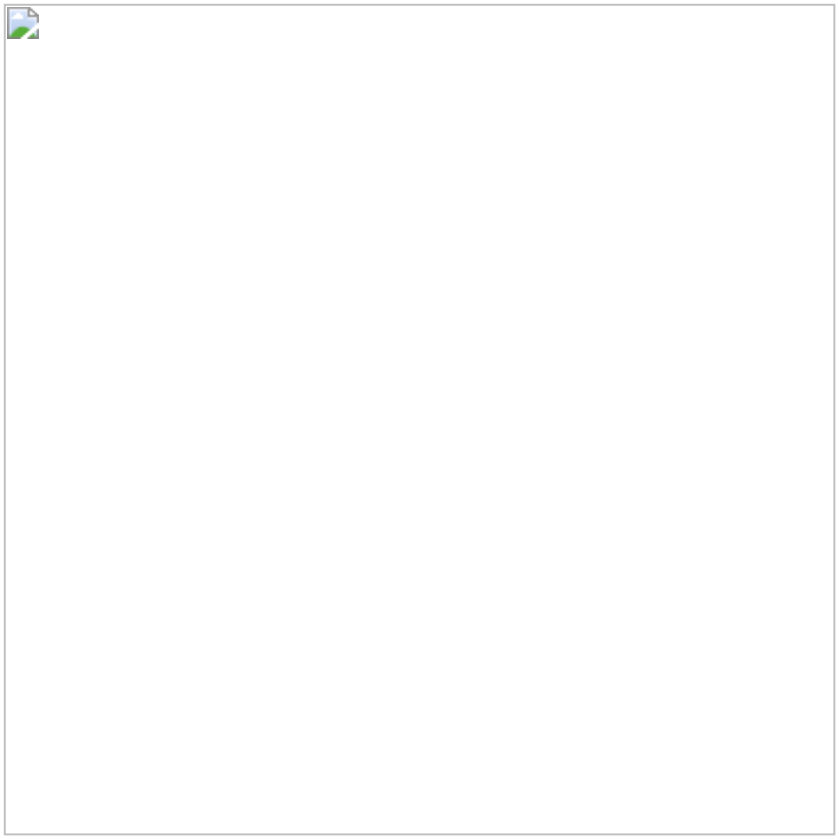
- Adversaries will meet resistance in leveraging stolen administrative credentials.
- Credentials from higher tiers should not be in memory on a lower tier machine.

[3] Denying all SMB communication between workstations

We leverage Windows firewall to DENY all inbound SMB communication to workstations.

We don't provide exceptions to this policy.

We use a simple Windows Firewall rule, distributed via Group Policy and applied to the workstations OU. The rule denies all inbound communication on ports 139 and 445.



> Note: We also recommend that you deny inbound WinRM and RDP to workstations and don't allow the machines to use LLMNR, Netbios or mDNS outbound.

The improvement:

- Lateral movement with SMB between our workstations is unlikely.
- Malware which spreads via SMB is also unlikely to move through our workstation fleet.

[4] Denying most SMB communication from workstations to servers.

We've audited the requirements of each of our servers and DENY SMB inbound to any that have no business need.

This single step reduces the overall risk in a very significant way. SMB-based lateral movement is now highly unlikely for the majority of our servers, and the remaining machines are designated as 'high risk' and will require additional controls and monitoring. Teams supporting

‘high risk’ servers accept that additional controls and constraints may apply to servers in this category.

Servers that *require* SMB in most Windows environments will be limited to Domain Controllers, File Servers, logging servers and a small handful of environment specific devices. It’s difficult to provide accurate numbers, but in our experience a reduction in exposed servers is likely to be greater than 90%.

The recommended implementation approach is similar to workstations. Link a “DENY SMB” policy at the Servers OU and use security group based filtering to prevent the small number of high risk servers from applying the policy.

> DENY the “apply group policy” right to the high risk servers group in the security ACL for the new SMB group policy.

The improvement:

- Significant reduction in surface area for SMB-based lateral movement.
- Reduction in the number of servers that require a compensating control makes it easier to tailor a solution to the server type and function.

[5] Prioritize Operating System upgrades for ‘high risk’ servers

SMB-based lateral *generally* starts by copying a payload to the remote target. The payload is then executed via one of a handful of techniques: Service Control Manager & WMI are common examples.

To provide an example:

The Microsoft Sysinternals utility psexec.exe deploys a binary to the Admin$ share on the remote machine. It then uses the DCE/RPC interface over SMB to access the Windows Service Control Manager API. That operation starts the PSExec ‘service’ on the remote machine and creates a named pipe that can be used to send commands to the system.

Many adversary tools also use this approach or substitute WMI in the code execution step.

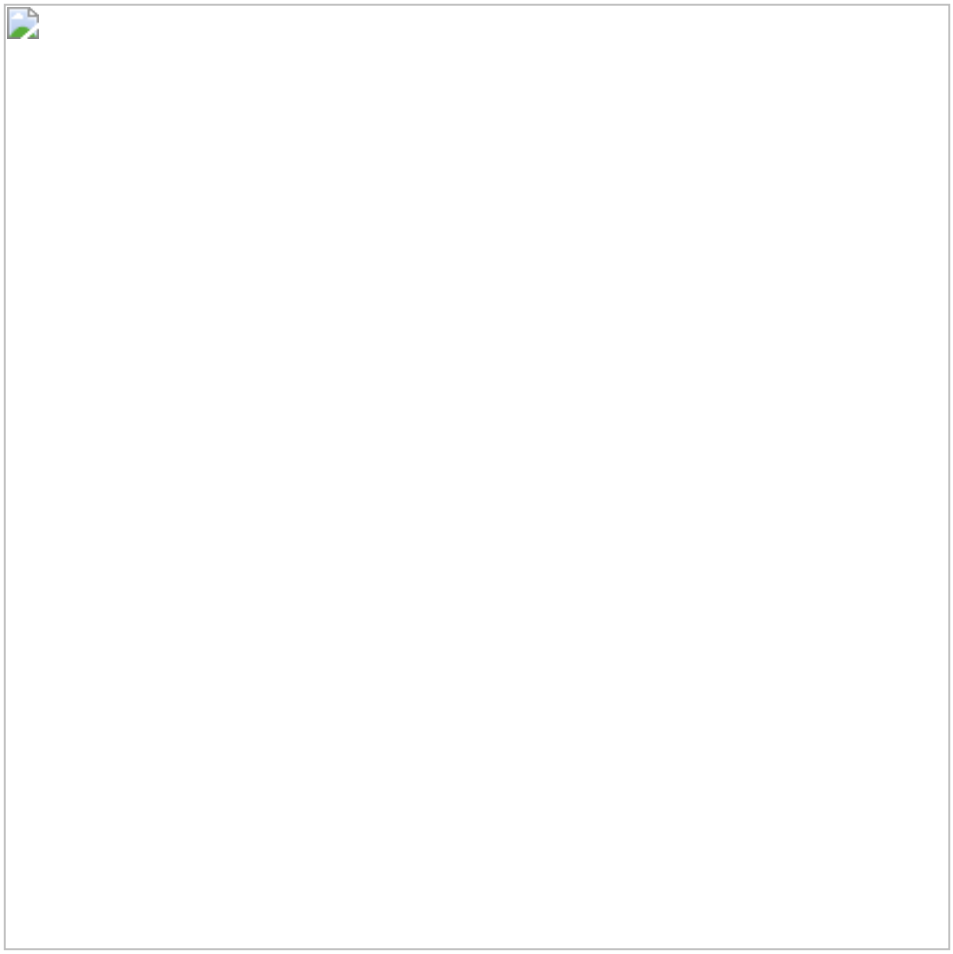
The summary of access required to make this work is:

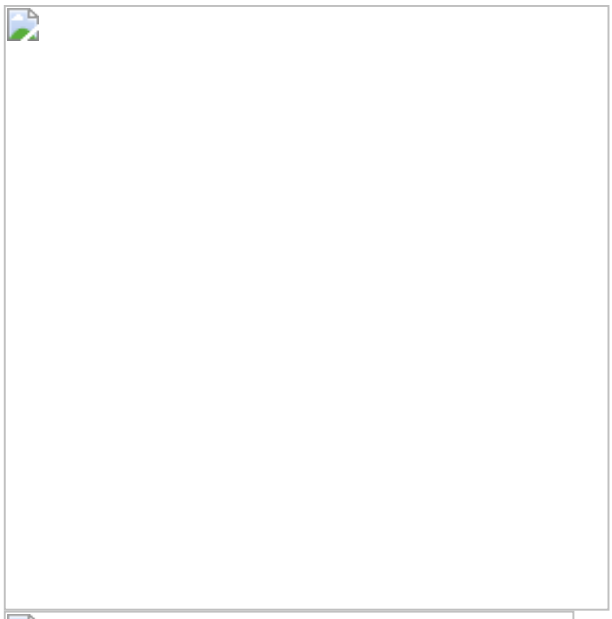
- TCP Port 139 or 445 open on the remote machine, i.e., SMB.
- Write permissions to a network shared folder.
- Permissions to create services on the remote machine: SC_MANAGER_CREATE_SERVICE (Access mask: 0x0002). or permissions to use WMI.
- Ability to start the service created: SERVICE_QUERY_STATUS (Access mask: 0x0004) + SERVICE_START (Access mask: 0x0010) — or permissions to start a process via WMI.

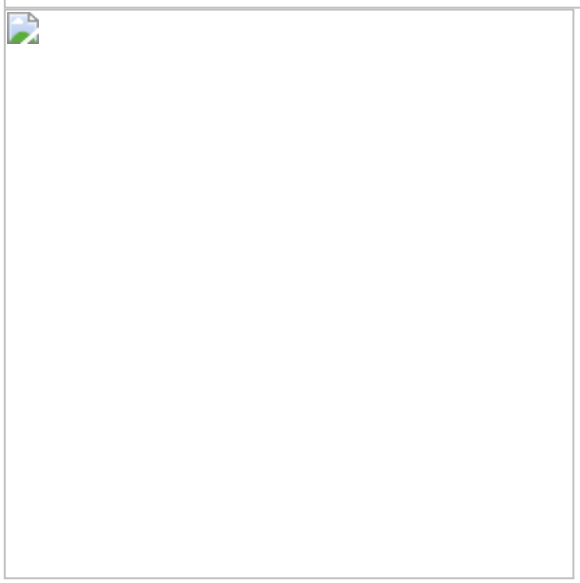
The challenge for defenders is that even in a well tiered environment, the list above are all permissions that a tier 1 administrator account would have, and with SMB, the adversary has an MFA bypass available allowing them to leverage stolen credentials.

The recommendation we've adopted is to adjust our corporate security policy to *require* that High Risk servers run the latest Windows Operating System.

Windows Server 2019 Defender will provide a significant improvement without configuring any additional control. This is because Defender is especially effective when a payload touches the disk. This will usually happen when the default SMB lateral movement approaches are attempted. Without additional effort on the side of the adversary, payloads from Cobalt Strike, Empire, and Metasploit are likely to be intercepted when copied to the disk of a Windows Server 2019 server.







The improvement:

- Default payloads will be destroyed by the Defender service.
- Alerting and Detection improvements when Defender eats a malicious payload.
- Opportunity to leverage additional modern OS controls in later steps.

Note: the recommendation is *not* to rush a 2019 upgrade/replacement for your entire fleet (although that would be nice). Instead, we've identified a subset of servers that still expose SMB in the previous step and classified them as 'high risk' — they're the urgent ones.

[6] Windows Defender Code Integrity rules

Windows Defender Code Integrity policies provide modern application whitelisting that works especially well when the role of a server is static and well understood. We combine this recommendation with the advice to deny SMB outright to most servers. Code Integrity is the primary compensating control for the small number of servers that have a business need to expose SMB to clients: our 'high risk servers'.

Windows Device Guard Code Integrity provides an 'Audit Mode', allowing administrators to deploy the technology for a period of time, then review the logs, adjust the policy, and repeat until a block operation is a high-fidelity event. Even without ever moving to the blocking phase, this provides a lot of value for the detection team.

> Matt Graeber's post is the go to for those wishing to use code integrity in their detection strategy: https://posts.specterops.io/threat-detection-using-windows-defender-application-control-device-guard-in-audit-mode-602b48cd1c11

Carefully implemented, code integrity policies provide an excellent compensating control for our servers that still require SMB to be exposed to clients. We've been fortunate enough to work directly with Matt Graeber and recommend his blog for detail. Our tips:

- Make heavy use of audit mode before attempting anything in blocking mode. You can iterate on audit mode policies until you no longer log any events (because you've added everything you need to the whitelist) and then leave the policy in place for a period of time to be sure.
- Start out with a specific server or service type you'd like to protect and perfect the process before attempting anything at scale. Domain Controllers are a reasonable choice because they have a single, well-understood role and shouldn't have much 3rd party software churn.

The improvement:

- Attackers are unable to run arbitrary code on high risk servers.
- Very strong control (but requires configuration time investment).

[7] Attack Surface Reduction rules

If you're operating in a blue team environment and haven't come across Attack Surface Reduction rules, Microsoft has a treat for you.

ASR rules are simple to deploy and incredibly effective, although not all the rules will be applicable for all environments. The majority of them offer audit mode to improve your ADS coverage, even if there is no desire to use block more.

> ASR rules are generally targeted more towards workstations than servers but there is still significant, low effort value available for defenders.

The current ASR rules are:

- Block executable content from email client and webmail
- Block all Office applications from creating child processes
- Block Office applications from creating executable content
- Block Office applications from injecting code into other processes
- Block JavaScript or VBScript from launching downloaded executable content

- Block execution of potentially obfuscated scripts
- Block Win32 API calls from Office macros
- Use advanced protection against ransomware
- Block credential stealing from the Windows local security authority subsystem (lsass.exe)
-
- Block untrusted and unsigned processes that run from USB
- Block Office communication application from creating child processes
- Block Adobe Reader from creating child processes
- Block persistence through WMI event subscription

The bolded one, “Block process creations originating from PSExec and WMI commands,” is especially interesting for some environments:

> “This rule blocks processes created through PsExec and WMI from running. Both PsExec and WMI can remotely execute code, so there is a risk of malware abusing this functionality for command and control purposes, or to spread an infection throughout an organization’s network.”

That’s a built-in control that will stop the PSExec-style lateral movement and provide excellent coverage against some of the other approaches that use WMI to trigger the code execution.

You can deploy ASR rules with group policy, and remember that even if you don’t want to *block* on these, you can put them in audit mode and use the events in your ADS’s.

> Note: The PSExec/WMI ASR kills SCCM and Intune. If your high risk servers are managed by those, you’ll have to consider the trade-off. In either case, you can leverage audit mode.

The improvement:

> OS-level protection against common SMB-based lateral movement.

[ALL] Combining the recommendations

Combining the recommendations above, we arrive at a baseline where:

- Credentials are restricted by tier, and stolen credentials from one tier provide limited value to an attacker who is seeking to improve their position in the network (MFA and Bastions are required).
- Very few servers in the environment allow network connectivity to the SMB service (port 139/445).
- The remaining servers are categorized as ‘high risk’ and have strong and simple-to-manage protection via Defender.
- ‘High risk’ servers leverage built-in lateral movement controls via attack surface reduction.
- ‘High risk’ servers prevent arbitrary code execution via code integrity policies.

Mitigations that didn't pan out for us

The following options were discussed and/or tested as potential improvements to our strategy. They either didn't work, or had significant downside.

Our reasons for each are discussed below:

[1] Deny network logon type for tiering violations

In mitigation [2] we recommended that teams configure "allow log on locally" in a way where tier restrictions are enforced by group policy.

We provided an example: '**runas /user:contoso\bob-t1 'psexec.exe \\tier1server.contoso.com cmd.exe'** where 'bob-t1' represents a stolen server administration account trying to execute psexec from a workstation, targeting SMB-based lateral movement to a server.

The control we've applied works with psexec defaults but as we mentioned, the workaround (build into adversary tools) is to force a 'network only' logon type.

There is a policy that says "Access this computer from the network" — more detail here. It's defined down in: Computer Configuration\Windows Settings\Security Settings\Local Policies\User Rights Assignment which sounds like it might provide value against network only-type logons.

It was suggested that maybe we could prevent our admins from doing the network only-style logon, while still allowing them standard logon.

Unfortunately, this option is not appropriate here. If you applied this setting to your servers then legitimate server admins will not be able to log on.

Cross it off the list.

> Note: You could still use this policy as an extra control to stop tier violations. You just can't use it to get additional granular control over legitimate users.

[2] Deny 'log on as a service' for administrative accounts

This consideration was based on one of the main approaches for SMB lateral movement: leveraging the service control manager to execute a payload.

It stands to reason that if we could somehow restrict server admins from interacting with the service control manager except via some break glass process, we could kill off this particular code execution approach.

The group policy is documented here. It's set here: Computer Configuration\Windows Settings\Security Settings\Local Policies\User Rights Assignment.

We didn't have success with this setting in our testing.

Microsoft also advise against it: "*We recommend that you not assign the Deny log on as a service user right to any accounts. This is the default configuration.*"

[3] Token filtering policies

There's a security configuration setting called "LocalAccountTokenFilterPolicy" that may provide some protection against lateral movement via local administrative accounts with the same password. However, we use LAPS in our environment and work with an assumption that no devices share local admin passwords. We capture LAPS password change events and have a good level of confidence in the solution.

Token filtering policies are interesting, though, and Will Schroeder did a great job of discussing the risks in this blog post.

We wanted to mention token filtering policies here because when discussing lateral movement they *should* come up. For this effort though, we are specifically targeting SMB and already have controls that would restrict pass the hash using local administrative accounts. Token filtering policies were not part of this specific defensive effort but if you are working through your lateral movement strategy they are an important consideration.

Token filtering policy STIG is here.

[4] Improved Service Control Manager ACL

We started out with an assumption that the Service Control Manager approach to getting binaries to run on remote systems was very common in SMB-based lateral movement, but through iterations with the red team, we realized that more often than not the WMI (over SMB) approach was preferred. With that in mind, we abandoned this control, but wanted to share our notes.

The hypothesis was that we could restrict our tier 1 administrative accounts from being able to create new services, then implement a process to add this ability back for the rare occasions on which an administrator would actually need to create a new service.

It's all documented here, but in brief — we can inspect the permissions for a service using:

```
sc sdshow scmanager
```

The result is a SDDL string like:

```
D:(A;;CC;;;AU)(A;;CCLCRPRC;;;IU)(A;;CCLCRPRC;;;SU)(A;;CCLCRPWPRC;;;SY)(A;;KA;;;BA)
(A;;CC;;;AC)S:(AU;FA;KA;;;WD)(AU;OIIOFA;GA;;;WD)
```

If you're not familiar with SDDL, it looks horrible, but it's actually reasonably straightforward to interpret. What you're seeing in the output above is both the DACL (security) and the SACL (auditing). You can break it up by separating on the D: and S: like:

```
(A;;CC;;;AU)(A;;CCLCRPRC;;;IU)(A;;CCLCRPRC;;;SU)(A;;CCLCRPWPRC;;;SY)(A;;KA;;;BA)
(A;;CC;;;AC)(AU;FA;KA;;;WD)(AU;OIIOFA;GA;;;WD)
```

We only care about the D: part for our goal.

Then you break up the individual access control entries. They're separated by brackets so (A;;CC;;;AU) is an ACE, (A;;CCLCRPRC;;;IU) is an ACE and so on.

Then, each ACE can be interpreted the same way:

- ACE type (allow/deny/audit)
- ACE flags (inheritance and audit settings)
- Permissions (list of incremental permissions)
- ObjectType (GUID)
- Inherited Object Type (GUID)
- Trustee (SID)

So using the first example: (A;;CC;;;AU) breaks out to:

(A;;CC;;;AU)

- Allow
- .
- Create All Child Objects
- .
- .
-

If we wanted to build a new ACE for the ACL, we'd construct it in the same way and add it to the string. You could set the configuration with:

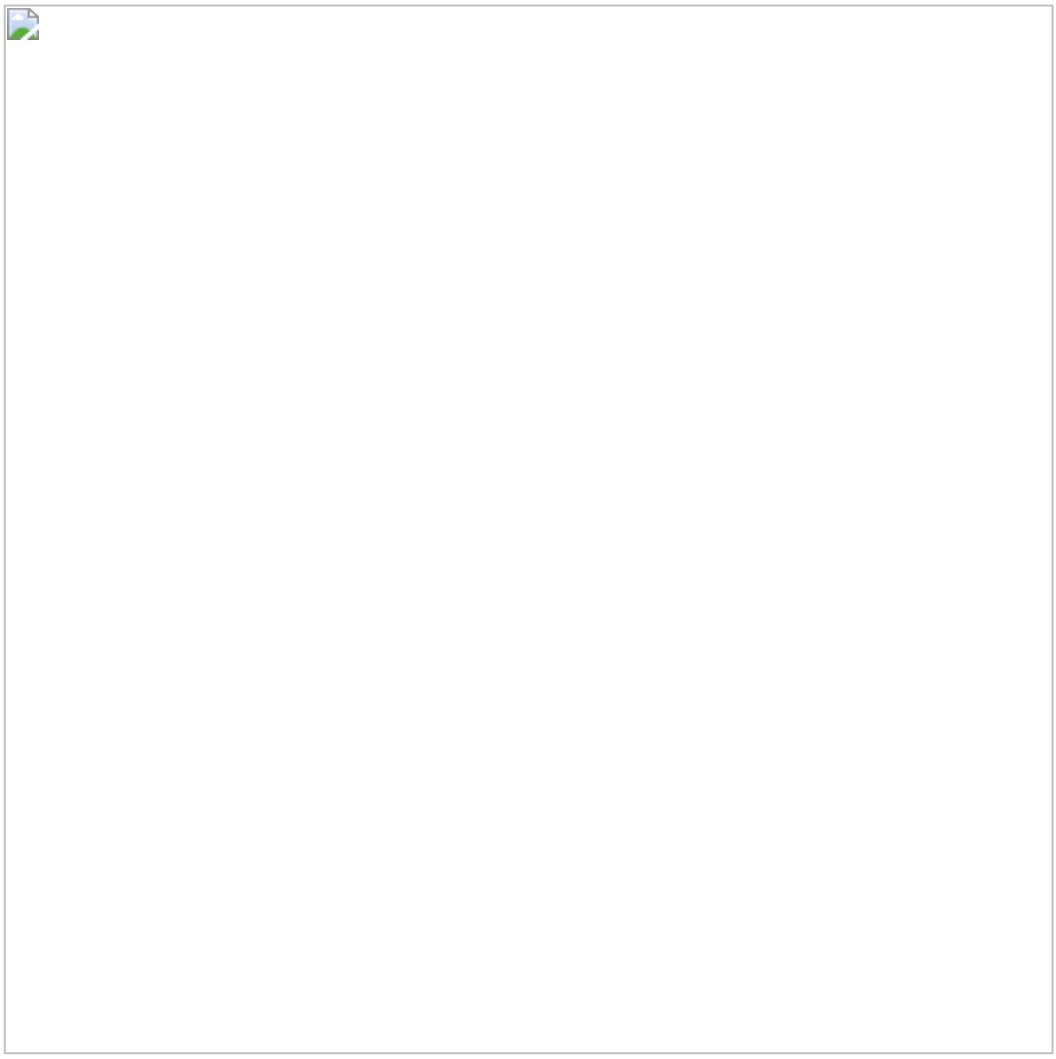
```
sc sdset scmanager NEW-SDDL-STRING
```

Our idea was to construct a new SDDL string where our "all-tier1-admins" group didn't have the 'create service' right and then deploy the new ACL to all our servers.

While it wasn't as complex as it first seemed, and did provide effective control against this specific type of SMB-based attack, we ultimately opted out because it provided no cover for SMB-based movement that used a different RPC approach.

[5] Local WMI Access restrictions

We considered modification to the WMI permissions for each server to restrict process creation using the common adversary approaches. There are a few ways to make the changes; the most direct is to use “wmimgmt.msc.” You could make adjustments via the security tab and potentially export configurations.



After spending a small amount of time here, we abandoned this line of research. We did so for the same reason we abandoned the service control manager changes. The level of complexity weighed against the incomplete coverage made it less compelling. We also realized that new processes would be required for break-glass and that we’d need to be very careful with the compatibility of applications and services, especially SCCM.

Ultimately, the juice wasn’t worth the squeeze while good alternatives such as Code Integrity policies existed.

[6] Advanced Windows Firewall configurations for all SMB traffic — IPSEC (null encryption)

We use this approach a *lot* already and highly recommend it for other use cases.

The premise is:

- Use Windows firewall to deny access to administrative ports (RDP for example).
- Then use some properties of the firewall
- DENY usually overrides ALLOW in Windows firewall.
- The one exception to that rule is IPSEC.
- If you use IPSEC you can override a DENY in a very granular way.
- (RDP example) DENY access to RDP for everyone, from everywhere, then configure an IPSEC policy for those that need RDP access to override that rule.
- The IPSEC policy can use Kerberos to further restrict access to specific users and groups.
- We use null encryption. We are leveraging the authentication component of IPSEC, but no encryption.

The reasons we decided not to take this approach for restricting SMB-based lateral movement:

- We'd still like administrators to be able to use SMB from bastion servers (achievable; but adds complexity).
- We can't configure a blanket DENY rule for SMB then override with IPSEC policy specific to our administrative accounts. We'd have to apply the override for all normal accounts that do need SMB access, and block the admins — so it's kind of the opposite of the model we are comfortable with.
- There's risk. There's potential for chicken and egg problems with the domain controllers and IPSEC, especially when considering GPO application may require SMB. Chances are the risks could be designed around for safe implementation, but given alternative controls were available, we decided not to pursue this route.

[7] Network-based tiering restrictions on a per service level

Our network team have the ability to get very granular with network traffic. We have ACEs on the firewalls that tie into active directory groups, and the team are able to inspect for service types in addition to traditional port-based restrictions.

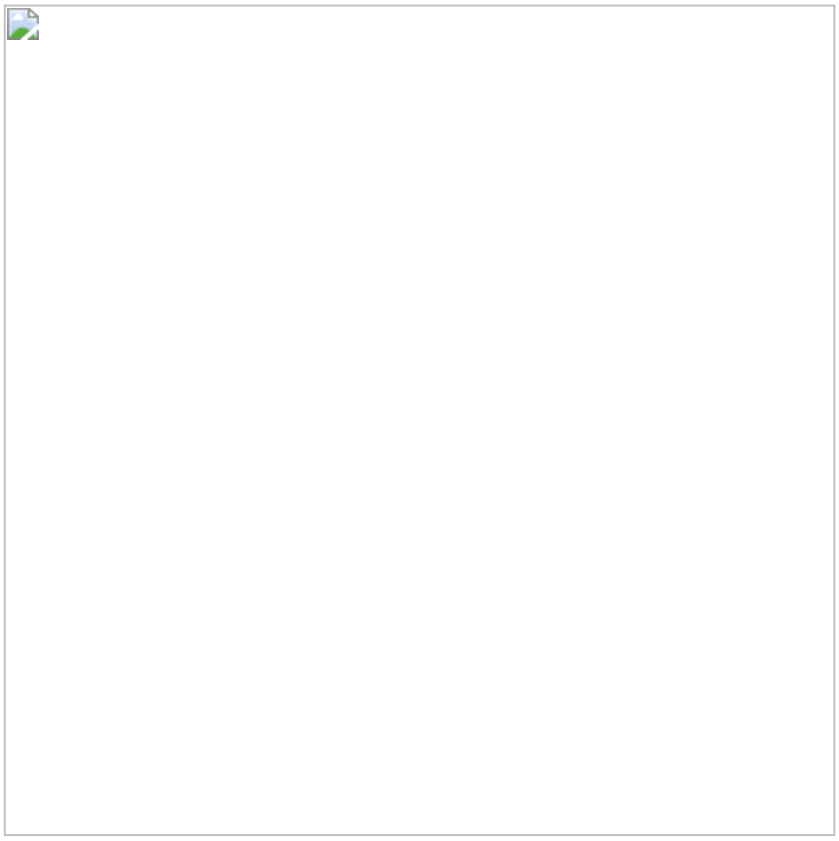
We found it difficult to accurately break the problem down in a way that scaled and responded well to change. With SMB in particular, we are usually dealing with legitimate credentials, and it is a legitimate service.

We *did* deny our tier 1 and tier 0 accounts from using SMB from any workstation network, to any server network. The challenge was the 'network login' type that we mentioned earlier in the psexec notes: if the adversary was able to perform network logon, the traffic appeared to originate from a non-administrative account and would PASS the ACL.

[8] Windows Firewall built-in Named Pipe rules

When we found this built-in setting, we thought we'd struck gold.

If we interpreted the rule the way it is written — “Remote Service Management (NP-In)” where NP is “Named Pipes” — we’d have an amazing level of protection against this problem for very little effort. (Remember that the majority of lateral movement via SMB works through Named Pipes).



Unfortunately, there is no special sauce in this rule.

It’s actually a port-based rule (445), with a fancy name describing why 445 might be needed. So if you switch it to “Block the connection,” you’ll be denying all SMB traffic — which is not our goal for all servers (and is already implemented on the others).

General SMB Advice

While not directly related to lateral movement, we felt it important to mention two extra points that are relevant to running SMB services safely:

- if at all possible: Ned Pyle provides all the reasons in post.
- Test, then for client and server on all devices. Microsoft provide advice .

Wrapping it up

We’ve spent a lot of time thinking about lateral movement. We work with the fundamental principle that administrative port access to servers is not allowed from our workstations at all, and that all administrative access should be via a bastion and require MFA.

For the longest time, SMB has provided a frustrating lateral movement gap in this security baseline that makes things easier for our adversary simulation team.

It's a cat and mouse game, but we're feeling confident that the changes in this post significantly raise the bar. We hope the lessons learned can be useful to others tasked with protecting their environments from adversarial lateral movement.

Author

Chad D.