How to own any windows network with group policy hijacking attacks

Author: Luke Jennings

For those of you that didn't make it to SyScan '15 last week, this is a blog post version of the presentation I gave about the vulnerabilities I found in group policy that resulted in Microsoft releasing **MS15-011 and MS15-014** in February. These bulletins resolve issues in Microsoft's group policy engine that allow remote code execution at SYSTEM level if an attacker can intercept network traffic from a domain-joined system.

The full process leading up to their discovery, along with exploitation details and accompanying video demonstrations, will be given. Additionally, the new security controls introduced by Microsoft will be discussed to assess how effective they are, show what attack scenarios they mitigate and which ones they do not. Finally, some additional guidance will be given on other mitigating controls that can provide additional protection will be discussed.

How does group policy work?

Group policy is Microsoft's core infrastructure for managing the configuration of both users and computers in an enterprise windows forest. Configuration settings can be grouped into group policy objects (GPOs) and these can be linked to various physical and logical groupings of computers and/or users. Alongside Microsoft Office, it is arguably the defining reason that Microsoft still dominate the business endpoint market. Some of its key selling points are summarised below:

- Allows highly flexible, centralised machine and user configuration over global enterprise networks
- Smart redundancy, designed to handle many sites/offices distributed over a global WAN

- It is key to enforcing security critical configuration settings across an enterprise

In a windows domain, there are domain members and domain controllers. Domain controllers store all GPOs and domain members communicate with domain controllers in order to determine which GPOs are relevant to them and to fetch the GPOs themselves. At a high level, there are four key services that allow this to happen:

1. **DNS** – Used to find the nearest domain controller
2. **RPC** – Used to establish a secure channel with the domain controller make various RPC calls
3. **LDAP** – Used to query the high level group policy configuration and which GPOs should be apply
4. **SMB** – Used to get the full GPO content for each applicable GPO

There are a huge number of configurable settings that can be set via group policy but the wealth of these end up as enforced registry changes on the endpoint. However, some notable exceptions to this are those that involve user accounts or group membership changes and file creation/modification. Different settings fall within different client side extensions (CSEs) and these CSEs will only be enforced on the client side if it has been indicated via LDAP that the relevant CSE is applicable for any given GPO. There are a large number of CSEs but some examples include security settings, registry settings, user accounts/groups, internet explorer zones etc.

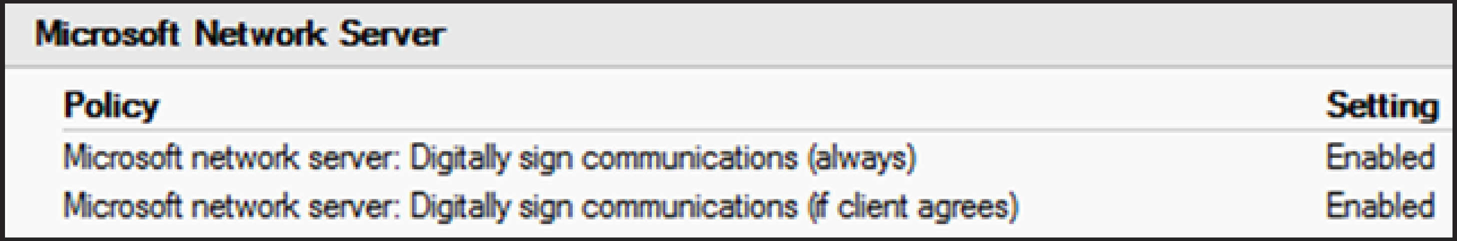
How can it be attacked?

First we will cover the situation prior to February 2015 (when MS15-011 and MS15-014 were released in response to these attacks) and look at what attack scenarios apply and how they could be exploited. The first important point to understand is that the sheer control group policy affords over system configuration means that if you can control group policy then you effectively have full SYSTEM equivalent control over the systems under it. Beyond that, we need to understand that each of the four key steps outlined earlier need to complete successfully in turn for the next one to begin. For example, if the RPC stage fails then we won’t reach the LDAP stage and so any attacks focused on later stages need to ensure the earlier stages complete.

The focus of this post is on the latter two protocol stages of the group policy update process, LDAP and SMB. These are the most interesting from a security perspective because LDAP is used to specify which GPOs and CSEs should apply to a host and SMB is used to fetch the individual GPOs including all the detailed configuration changes they contain. Control over either of these stages would give us significant control over the domain member that is fetching group policy.

So what security controls are in place to protect LDAP and SMB communications? Well the key concern here is that the domain member verifies the identity of the domain controller to ensure it is a getting its group policy from a legitimate source and that integrity protection is in place to ensure the data can not be tampered with in transit. These are principally handled via authentication within the protocols plus integrity protection in the form of LDAP signing and SMB signing. We will focus on SMB from this point forward as this would give the most flexible control over individual configuration options due to it being used to fetch the individual GPOs themselves.

There are configurable settings in group policy for controlling security critical options related to SMB signing. An excerpt of the “default domain controllers” policy is given below:



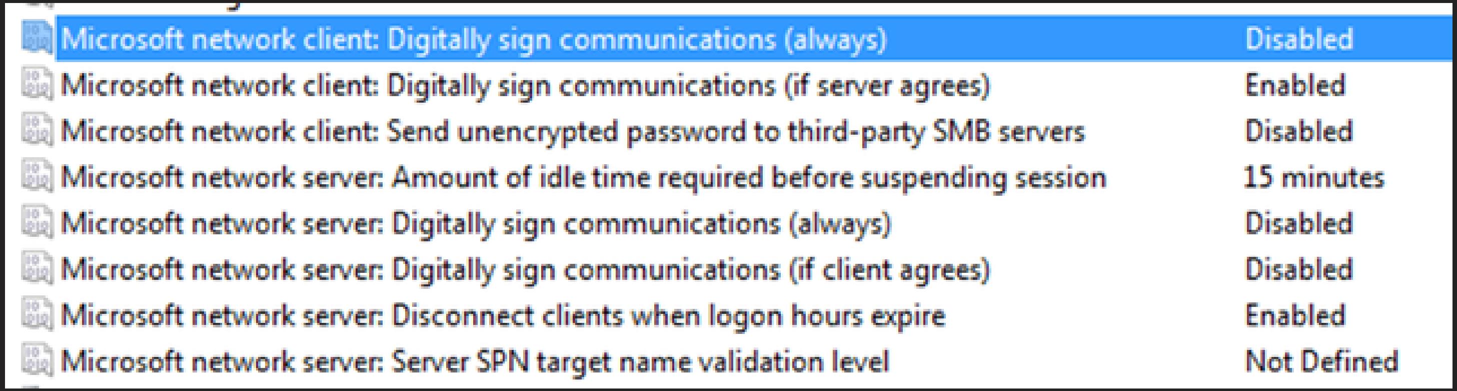
Microsoft Network Server

Policy	Setting
Microsoft network server: Digitally sign communications (always)	Enabled
Microsoft network server: Digitally sign communications (if client agrees)	Enabled

As you can see, the default behaviour for domain controllers is to require SMB signing. This is a good control to have in place as we would not want SMB connections pulling down GPO information from domain controllers without integrity protection in place. This was alluded to in one of Microsoft's **technet articles** from back in 2010 with the following quote:

"SMB signing is available in all currently supported versions of Windows, but it's only enabled by default on Domain Controllers. This is recommended for Domain Controllers because SMB is the protocol used by clients to download Group Policy information. SMB signing provides a way to ensure that the client is receiving genuine Group Policy."

However, that does not paint the full picture as it tells us nothing about the client. The relevant group policy options for specifying SMB signing on the client side are not specified in the "default domain policy", which is applied to all domain members. Consequently, we need to look at what the default local configuration is. An excerpt of this showing the relevant settings is given below:



Microsoft network client: Digitally sign communications (always)	Disabled
Microsoft network client: Digitally sign communications (if server agrees)	Enabled
Microsoft network client: Send unencrypted password to third-party SMB servers	Disabled
Microsoft network server: Amount of idle time required before suspending session	15 minutes
Microsoft network server: Digitally sign communications (always)	Disabled
Microsoft network server: Digitally sign communications (if client agrees)	Disabled
Microsoft network server: Disconnect clients when logon hours expire	Enabled
Microsoft network server: Server SPN target name validation level	Not Defined

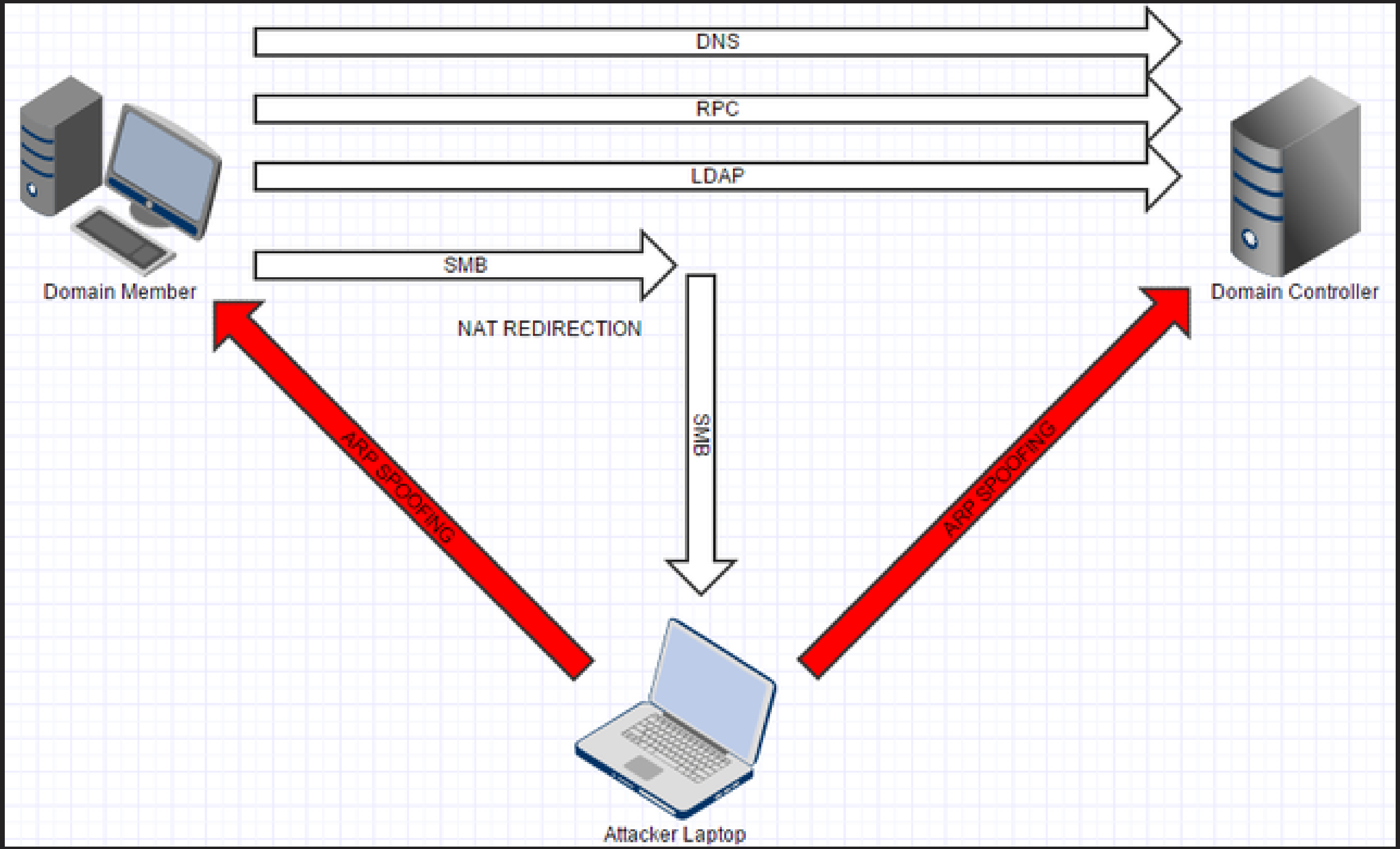
As can be seen, SMB signing is not set to be a requirement on the client side. This means that domain members acting as an SMB client will negotiate SMB signing but will not require it if it is not supported.

Defeating SMB Server Signing

If the domain member does not require SMB signing but the domain controller requires SMB signing then how do we attack it? If we were in a position intercepting network traffic then we could either allow signing to be negotiated and then the connection would go on fine but be protected or we could prevent signing from being negotiated but the server would terminate the connection. However, we can avoid the domain controller SMB service altogether to eliminate the problem. The key points outlining this are given below:

1. We allow DNS, RPC and LDAP traffic to all pass through from the domain member to the domain controller unhindered
2. We redirect SMB traffic to our own malicious SMB server via NATing rules or similar
3. The “real” SMB service on the domain controller is never involved so we can force SMB signing to be disabled on our own malicious SMB server
4. The domain member SMB client happily negotiates no signing and carries on with unprotected SMB communications talking to a malicious server

The diagram below gives an idea of how this attack would work in practice using ARP spoofing as the example traffic interception technique:



Getting a shell

What we have discussed so far is nice in theory but how do we practically exploit it to get a shell on the target system in practice? We could define login scripts, we could define new user accounts or abuse other settings to gain control. However, we need to remember that only the CSEs indicated to be active for a given GPO during the LDAP phase will be applied. The CSE for user management may not be in use, login scripts may not be in use and may not get us privileged access anyway etc and so our ideal exploitation technique would depend only on CSEs in use by default and on options that would get us SYSTEM-level access.

It turns out that the security settings CSE is applied by default and allows us to control arbitrary registry settings. If we make use of the GPO section for this and define our own custom registry settings then we have full control over the registry on the domain member, which is very powerful. There are various ways we could use this to get code execution such as defining “Run” keys, creating new services, defining new authentication providers etc. However, during my testing I settled on **AppInit DLLs** for a proof of concept.

AppInit DLLs are DLLs that are loaded into any process when it starts. My reason for choosing these was that no matter what, even on an inactive, untouched server that does not reboot, sooner or later a new process will be created in the background and load our DLL and there are also likely ways we can trigger a new process easily anyway. In this case, we can specify a meterpreter DLL payload using a UNC path on an SMB server we control and then next time a new process starts we will get a shell. If the system is running RDP then we can use rdesktop to connect, which will spawn a new winlogon.exe process as SYSTEM, loading our DLL payload and giving us a meterpreter shell as SYSTEM instantly. We will see a video of this exploit in action later but for now we will move on to considering a more secure configuration.

(Breaking) Secure Configuration

So at this point we have seen that the default configuration of a domain is vulnerable to man-in-the-middle (MITM) attacks against SMB when domain members fetch group policy. This is the basis of what eventually became MS15-011. However, this is due to a poor default configuration that can be resolved. What happens when you upgrade the configuration such that SMB signing is required by the clients?

In this instance, I found a direct security control bypass vulnerability. I found that it was possible to deliberately corrupt the SMB communications when the security settings were being fetched, such that SMB signing would fail and the connection would be terminated. Now intuition would say that due to the failed application of group policy, the domain member would just remain at its existing configuration. However, it turned out that this was not the case and that as a product of the failure the group policy engine on the domain member would revert back to default configuration. Since the default local configuration does not require SMB signing, the domain member would revert itself back to an insecure state and then the original attack would apply as a second stage on the next group policy refresh. This is what became MS15-014.

Hardened Configuration Exploit Process

Now we will summarise the full exploit process for exploiting a hardened environment, along with an exploit video to demonstrate it:

1. ARP spoof domain member and domain controller
2. Allow all protocols to pass through fine (DNS, RPC, LDAP and SMB)
3. Wait until the security settings response comes back via SMB

4. Corrupt the response to cause the domain member SMB signing requirement to revert
5. Modifying our NAT rules to redirect future SMB traffic to our own malicious SMB server
6. Domain member will then fetch security settings containing our malicious AppInit DLL settings
7. We make an unauthenticated RDP connection to the domain member to trigger a new SYSTEM winlogon.exe process
8. Our handler receives a new meterpreter shell running as SYSTEM

The following video demonstrates this exploit in action exploiting both MS15-011 and MS15-014 in a staged attack to compromise hardened environments:

User vs Machine Attacks

Group policy contains separate settings for computers and users. Computer settings will always apply to the computer itself but user settings will apply depending on the user that logs on. Until now we have only really been considering computer settings.

After discovering the previous two attacks, I considered what would be possible if they were fixed by Microsoft and that a network was configured securely and so I began looking at SMB signing in more detail. In the case of NTLM authentication, SMB signing uses a key derived from the password of the user account used for authentication. The interesting finding I made here was that, whilst the machine settings use the machine account to access SMB, the user settings actually use the user's account.

Machine accounts by default use secure, randomly generated passwords and so we should not expect to be able to acquire or crack this password without already having gained SYSTEM access in the first place. This makes it a sensible account to be used as the basis of fetching group policy updates and should make the SMB signing mechanism strong. However, an ordinary user account may have a weak password or we may have discovered it through other means e.g. social engineering. In a privilege escalation scenario, it may even be our own account that is a low privileged domain account without any administrative control over domain members that it can login to. The key point here though is that if we know the password of a user logging into a system that we can intercept network traffic for then we can calculate SMB signatures correctly when user settings are fetched and so we can control the user settings part of group policy applied when they login.

The question then becomes whether this poses a problem or not. What is the real impact of controlling user settings for group policy? Well in order to exploit the issue we need to know the user's password and so intuition would say it would not represent an issue as controlling a user's own configuration is redundant when we already have access to everything they do anyway. However, it turns out that despite being "user" settings, they are actually a lot more powerful than that. There are CSEs for them that include the ability to add new user accounts or define arbitrary registry keys including within the SYSTEM hive and therefore controlling user settings can still be used to gain SYSTEM access on a domain member. The implication of this is that if you know the password for a low privileged user account logging in to a system that you can intercept network traffic for then you can use that to gain SYSTEM privileges.

There are probably two primary scenarios that this attack would apply to:

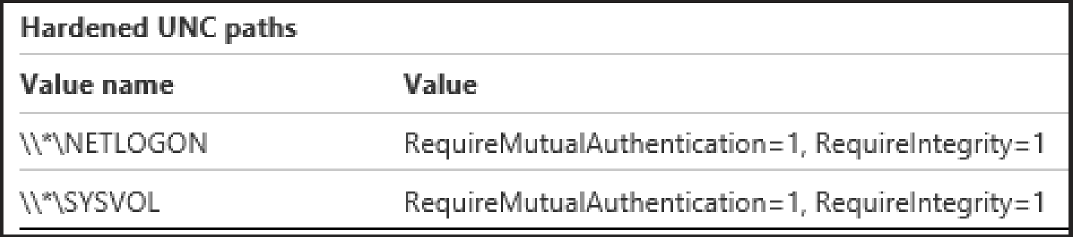
1. You have a low privileged account and can use it to login interactively to physical workstations or virtual desktop farms that you can intercept network traffic for. In this instance you could escalate your privileges on them to SYSTEM.
2. You are intercepting traffic on a large subnet full of laptop/desktop users and you know the passwords for a small number of uninteresting, low privileged users. However, you can use this knowledge to get full SYSTEM access to their endpoints, which may prove much more valuable to you.

The following video demonstrates exploitation of this issue:

Microsoft's Response – MS15-011 and MS15-014

In February 2015, Microsoft introduced two security bulletins to address the vulnerabilities I reported. MS15-011 was not a bug fix but instead introduced an entire new configurable security control set known as "hardened UNC paths", designed explicitly to thwart these types of attacks and more. They provide much more flexible and secure configuration over what happens when a UNC path is accessed than could be achieved previously. It is now possible to specify on a per-UNC path basis (with optional wildcards) what the security requirements for the connection are. Windows will then ensure it picks a transport mechanism that can meet those and ensures they are applied.

Microsoft then give two recommended rules to configure as a minimum in a domain environment to protect against the attacks outlined before. These are shown below and essentially state that if the NETLOGON or SYSVOL shares are accessed on any domain controller that the OS should ensure it uses a transport and authentication method that allows mutual authentication and integrity protection and that those controls are enforced.



Hardened UNC paths	
Value name	**Value**
*\NETLOGON	RequireMutualAuthentication=1, RequireIntegrity=1
*\SYSVOL	RequireMutualAuthentication=1, RequireIntegrity=1

This is actually further than I expected Microsoft to go and is a welcome step forward as they could have other potentially useful applications too. The Microsoft recommended settings shown above will also effectively disallow NTLM as an authentication mechanism for group policy updates as it does not strictly allow mutual authentication and so it will enforce kerberos use in general. This new security control when combined with the configuration above successfully protects against the first attack we outlined.

In contrast, MS15-014 was a straight bug fix. There are no new security controls to configure here, it simply prevents the group policy processing engine from reverting back to the default local configuration when it encounters a failure during the retrieval of the security settings, which is intended to prevent the second attack we considered. I tested my original exploit against the patched version and sure enough the SMB signing settings remained in a secure state and so it seems this patch is effective.

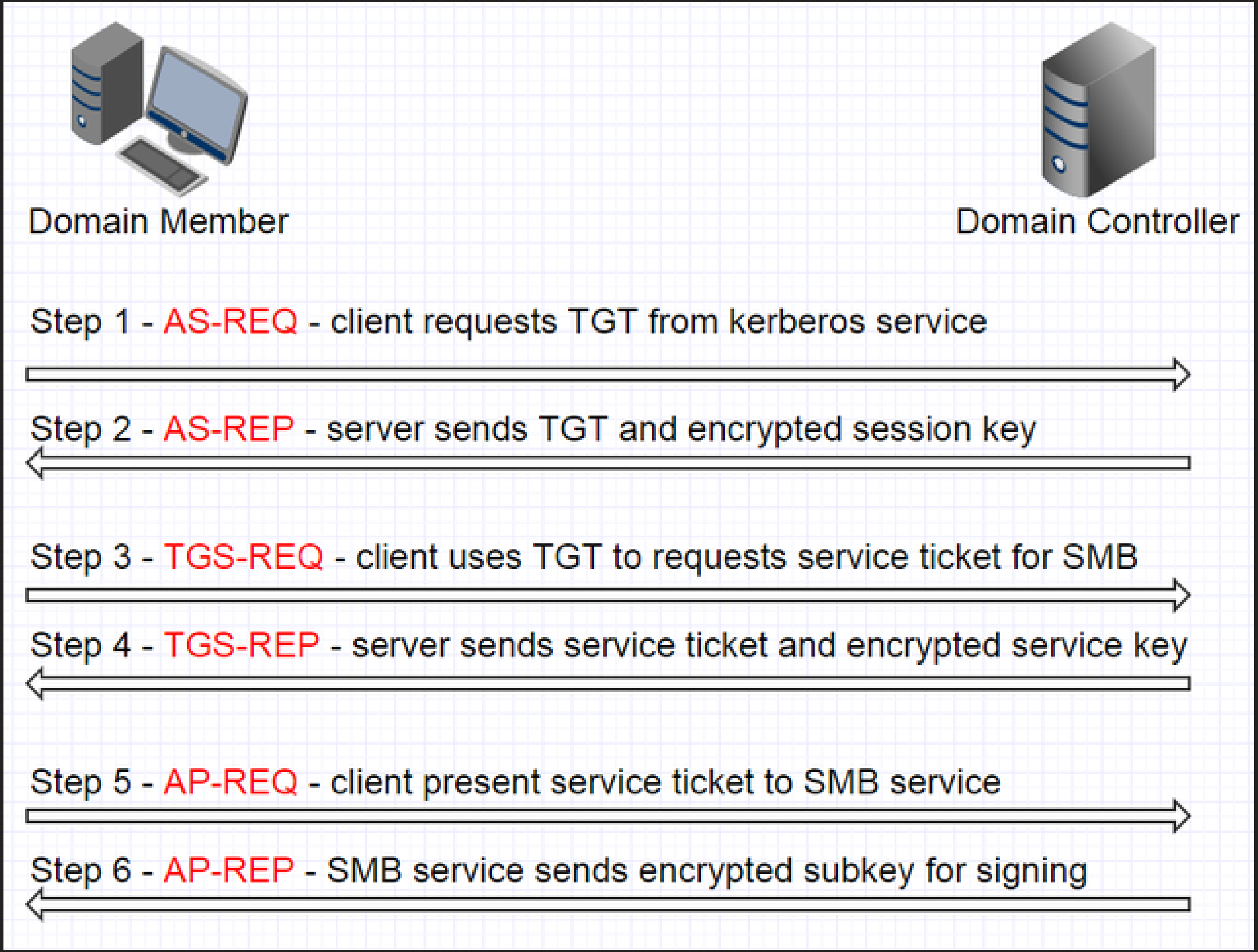
What happened to "User settings" post-patch?

Microsoft are certainly aware of this exploit scenario as it was one of the three issues I originally reported but it was also the one I most felt may end up being "by design". Nothing in the security bulletins for MS15-011 or MS15-014 specifically mention this issue and it is not immediately obvious how hardened UNC paths would resolve this issue. My testing revealed that the user account was still used for authentication when fetching user settings, not the machine account, meaning the underlying issue is still there. So the question is does this exploit scenario still apply?

Our previous exploit technique redirected the SMB traffic to our own malicious SMB server, where we also had a user configured with the same username and password. The domain member would negotiate SMB signing and use NTLM authentication and our server would know how to calculate the signature because it would know the password of the user account, which is used to derive the signing key. However, with properly configured hardened UNC paths, the mutual authentication requirement means NTLM will not be used and kerberos will be enforced. How does this change the scenario?

Decrypting Kerberos Packets

We won't be able to use the same exploit technique as before if kerberos is in use. Our malicious SMB server will not be able to authenticate itself to the domain member as it will not have the necessary keys to decrypt the service tickets that the domain member will supply to it. Additionally, the SMB signing keys are not a simple derivation of the user password when kerberos is used, they are generated by the SMB server and returned encrypted with the shared service key between the domain member and SMB server such that the domain member knows the signing key too. So is this an effective control? To understand that, we need to consider how a kerberos exchange works. The following diagram gives a simplified view of the process involved when a domain member wants to access the SMB service on a domain controller:


Domain Member
Domain Controller
Step 1 - AS-REQ - client requests TGT from kerberos service
Step 2 - AS-REP - server sends TGT and encrypted session key
Step 3 - TGS-REQ - client uses TGT to requests service ticket for SMB
Step 4 - TGS-REP - server sends service ticket and encrypted service key
Step 5 - AP-REQ - client present service ticket to SMB service
Step 6 - AP-REP - SMB service sends encrypted subkey for signing

The AS-REP (step 2) response contains a session key encrypted with a key derived from the user's password. If we are monitoring kerberos packets and know the user's password then we can decrypt this. The TGS-REP response (step 4) then contains a service key encrypted with the session key from before. If we have been keeping track of decrypted session keys then we will be able to decrypt this too. Finally, the AP-REP response (step 6) contains a sub key (used for SMB signing) encrypted with the service key from the TGS-REP response. If we have been monitoring and decrypting all the correct packets then we should be able to decrypt this sub key too. If we can eventually derive this sub key then we have everything we need to make arbitrary modifications to SMB packets and recalculate the signatures. Therefore, it seems the user settings exploit scenario still applies. To summarise, the following key steps apply in a post-patch, securely configured hardened UNC paths world:

1. We monitor all kerberos exchanges and decrypt all AS-REP packets encrypted with user passwords we know to obtain session keys
2. We decrypt any TGS-REP packets we can with the session keys we have collected to obtain service keys
3. We decrypt any AP-REP packets we can with the service keys we have collected
4. We use the sub keys we have obtained to make malicious changes to group policy information in SMB read responses and dynamically recalculate the SMB signatures on the fly
5. We get our SYSTEM shells

The following video demonstrates this issue being exploited on a patched system with securely configured hardened UNC paths.

One important caveat to mention at this stage is that user settings are a bit less flexible than computer settings. Earlier we discussed how we used the security settings CSE for exploitation as this CSE would always apply no matter what was configured for a GPO and are forcibly re-applied periodically too. With user settings, we don't have the same level of power by default unless certain more interesting CSEs are configured. In the exploit video above I had specifically configured a registry setting in the user settings GPO of the default domain policy such that the CSE for that applied and then my exploit modified the contents of the resulting XML file that was retrieved in order to inject a malicious AppInit DLL setting. In the default case though, this CSE would not be configured.

However, something we have not looked at in much detail yet is LDAP. I mentioned previously that it is used to obtain the GPOs that should apply to a system or user (uniquely identified by GUIDs) along with which CSEs are enabled for them and then SMB is used to retrieve the detailed GPO settings for each of these. When user settings are fetched, the user account (rather than the machine account) is also used to authenticate to LDAP using kerberos. Consequently, a very similar process to before is conducted to negotiate a signing key for LDAP. By monitoring and decrypting kerberos packets, we should be able to derive this key in the same way and make arbitrary changes to the LDAP packets. This would mean that we could specify to enable whatever CSEs we are interested in using for our exploit and as such ensure that we can exploit this issue in the default case. Whilst I haven't got as far as writing the code to verify this 100%, there is no reason why this should not work in exactly the same way as the SMB dynamic signature recalculation demonstrated practically in the video above.

Alternative security controls

All of these attacks are MITM attacks and so generally rely upon traffic interception techniques in order to conduct them. The most common and viable of these are layer 2 based, such as ARP spoofing, which is what was used in all exploit demonstrations above. There are a range of standard prevention and detection techniques for the various traffic interception attacks out there that are well worth investigating to provide additional protection against the attacks outlined in this blog post, as well as a range of other attacks that are dependent on traffic interception.

Conclusion

This concludes a pretty long and complicated blog post but I hope it has been interesting. Below, I'll summarise some of the key take home points from all of this:

- Prior to February 2015, all enterprise windows networks were vulnerable to MITM attacks that would allow an unauthenticated attacker to gain SYSTEM privileges on any domain member. All versions of windows from XP/2003 to 8.1/2012R2 were vulnerable.
- MS15-011 and MS15-014 introduced new security features and bug fixes to protect against these attacks in the form of a new security control called "hardened UNC paths".
- Microsoft have no intention of fixing XP/2003 and so these OS versions remain vulnerable.
- Vista/2008 onwards are still vulnerable in their default state as hardened UNC paths are not default. You need to configure them

explicitly to Microsoft’s recommended configuration and apply it across your entire estate using group policy.

- Even on Vista/2008 onwards, user settings group policy can be exploited if you know a user’s password to conduct a form of privilege escalation to gain SYSTEM on domain members. Microsoft have shown no intention thus far of providing a control to protect against this.
- All exploit scenarios rely on MITM attacks and so existing controls to prevent and detect traffic interception attacks are a good way to provide additional protection against these issues. This is particularly important for XP/2003, which Microsoft have not patched, or if you are particularly concerned about the user settings related privilege escalation style attacks, for which Microsoft have provided no direct protection.