Black Hat USA 2014: Why You Need to Detect More Than PtH

Matt Hathaway
matthew_hathaway@rapid7.com

Jeff Myers
jeff_myers@rapid7.com

Introduction

It does not matter where or how the attack started – while many attacks start with credentials, at some point all attacks look like an insider. On this premise, we believe that reducing the effectiveness of known attack techniques is as important as ever. Practitioners need to educate users, reduce the use of administrative privileges in an organization, actively avoid RDP, and do as much as possible to eliminate NTLM authentications. In spite of the progress Microsoft has made in recent years to mitigate known attacks like Pass-the-Hash (especially in Windows 8.1) this threat has not been eliminated.

We wrote this paper with no intention of introducing new attacks or expanding on the excellent Pass-the-Hash presentations over the last several Black Hat events. This is a defensive guide providing a series of steps necessary to make detection achievable for the incident response team. It is wholly intended to highlight where to look and what to look for so that compromised credentials can be detected.

Mitigation is Just That

Every security professional has an example fresh in mind: the new CFO that gives his assistant all of his passwords, the development team that simply cannot function without local administrator privileges, the domain administrator with the best ticket close rate in company history because he breaks proper account use policy if it means the ticket can be marked 'closed' an hour earlier. After speaking with security teams for a few years, we have learned that there is always a new reason for an exception to be made.

Guides like the "Mitigating Pass-the-Hash (PtH) Attacks and Other Credential Theft Techniques" are helpful, but not comprehensive. One must assume that some percentage of attacks will get through. *Taking a page from anti-fraud strategy used by the largest banks in the world, information security teams need to combine various mitigation and prevention tools to reduce the likelihood and impact of a breach with fast detection for the now-reduced level of attacks that are successful.*

How Not to Detect Stolen Credentials

While working with various organizations to detect compromised credentials and penetration testers to simulate attacks in which they are used, we started in a logical place: Microsoft Active Directory Security Logs. Since Microsoft has nearly monopolized the LDAP space, it provided a central collection point for account details and network authentication that only left the challenge of analyzing log messages. The naiveté of our initial assumptions was revealed when we worked with penetration teams experienced in evading detection who managed to leave next to no sign of network access.

These attacks can be found in the domain controller logs, specifically authentication between account and asset, and nearly all domain-level administrator actions. There are multiple indicators from existing accounts accessing new assets and administrative activities like privilege escalation that, when found, will leave an uncomfortable feeling in a responder's stomach.

However, the data in these logs cannot be relied upon to consistently detect an attacker using compromised credentials (or hashes) to move laterally through a network. The account used to originate the authentication never gets logged and the depth of the details of machine-to-machine authentications

is simply that they were “network” in nature. In addition, any local accounts that authenticate to an asset will forever lie in a gaping blind spot.

Richer Data Sources Needed

For the reasons explained in the previous section and the ultimate goal of detecting compromised local accounts and Pass-the-Hash attacks, we moved to the endpoints. Black Hat 2013’s “The Admin’s Revenge” talk ended with some great examples of the data available if using this valuable source, but every organization we quizzed while building this detection has PowerShell disabled on the vast majority of their assets. Based on that and some internal expertise, we decided to lean on Windows Management Instrumentation (WMI) as the technology that would give us access to the host-based event logs.

Once a means to collect the data was devised, it was time to work on the log analysis that had proven ineffective on the domain controller. To illustrate the difference in event sources, we will show a series of examples of what the logs contain on the domain controller, source host, and target host.

Remote Desktop Protocol (RDP)

Let us start with an example of host-to-host authentication commonly used in organizations to improve productivity. This is the associated raw entry for an RDP authentication that is logged on the domain controller:

```
- <Event xmlns="http://schemas.microsoft.com/win/2004/08/events/event">
- <System>
  <Provider Name="Microsoft-Windows-Security-Auditing" Guid="{54849625-5478-4994-A5BA-3E3B0328C30D}" />
  <EventID>4624</EventID>
  <Version>0</Version>
  <Level>0</Level>
  <Task>12544</Task>
  <Opcode>0</Opcode>
  <Keywords>0x8020000000000000</Keywords>
  <TimeCreated SystemTime="2014-06-17T19:39:33.444811300Z" />
  <EventRecordID>3471628</EventRecordID>
  <Correlation />
  <Execution ProcessID="460" ThreadID="1692" />
  <Channel>Security</Channel>
  <Computer>DC-01.testdev.com</Computer>
  <Security />
  </System>
- <EventData>
  <Data Name="SubjectUserSid">S-1-0-0</Data>
  <Data Name="SubjectUserName">-</Data>
  <Data Name="SubjectDomainName">-</Data>
  <Data Name="SubjectLogonId">0x0</Data>
  <Data Name="TargetUserSid">S-1-5-21-3971006398-2356616383-2175817166-1184</Data>
  <Data Name="TargetUserName">bob</Data>
  <Data Name="TargetDomainName">TESTDEV</Data>
  <Data Name="TargetLogonId">0x49c7ace</Data>
  <Data Name="LogonType">3</Data>
  <Data Name="LogonProcessName">Kerberos</Data>
  <Data Name="AuthenticationPackageName">Kerberos</Data>
  <Data Name="WorkstationName" />
  <Data Name="LogonGuid">{4441712D-E78E-F221-C81C-D6C95A0CB0B4}</Data>
  <Data Name="TransmittedServices">-</Data>
  <Data Name="LmPackageName">-</Data>
  <Data Name="KeyLength">0</Data>
```

```
<Data Name="ProcessId">0x0</Data>
<Data Name="ProcessName">-</Data>
<Data Name="IpAddress">10.1.102.51</Data>
<Data Name="IpPort">49804</Data>
</EventData>
</Event>
```

In the rest of the examples, we will focus only on the important data fields to avoid turning this paper into a word search. There are four relevant events logged for this RDP authentication, with the useful information shown in the table below:

Code	Target User Name	Target Domain Name	Workstation / Service Name / Logon Type	IP Address
4776	bob		LABCLUB2-1	
4768	bob	testdev.com		::ffff:10.1.102.51 (target)
4769	bob@TESTDEV.COM	TESTDEV.COM	LABCLUB2-2$	::ffff:10.1.102.51 (target)
4624	bob	TESTDEV	3 - Network	10.1.102.51 (target)

The message codes mean:

- 4776 - The domain controller attempted to validate the credentials for an account
- 4768 - A Kerberos authentication ticket was requested
- 4769 - A Kerberos service ticket was requested
- 4624 - An account was successfully logged on

This obviously offers some context, but there are some missing pieces. For comparison, here are the relevant messages on the target host:

Code	Subject User Name	Subject Domain Name	Target User Name	Target Domain Name	Logon Type	Workstation Name / Target Server Name	IP Address
4624			bob	TESTDEV	3 - Network	LABCLUB2-1	
4648	LABCLUB2-2$	TESTDEV	bob	TESTDEV		localhost	10.1.102.53 – source
4624	LABCLUB2-2$	TESTDEV	bob	TESTDEV	10 – Remote Interactive	LABCLUB2-2	10.1.102.53 – source

And the same on the source host:

Code	Subject User Name	Subject Domain Name	Target User Name	Target Domain Name	Target Server Name	Target Info
4648	alice	TESTDEV	bob	testdev	labclub2-2.testdev.com	labclub2-2.testdev.com

The new message code:

- 4648 – A logon was attempted using explicit credentials

To quickly summarize, the domain controller tells us that someone on *LABCLUB2-1* successfully authenticated to *LABLCLUB2-2* with the *bob* account, but there is no context of which account initiated the authentication and it is unclear that it is a remote interactive session. Whereas, the target host logs inform us that the authentication was type *10* and identifies the source host's IP address, and the source host tells us that the authentication went from the *alice* account being logged on locally to the *bob* account on *LABCLUB2-1*.

User Account Control (UAC) Prompt – Local Administrator

In the RDP scenario, domain accounts were used to show how much richer the host event logs are when compared to their centralized counterparts. Now, to truly understand the need for event logs on host systems, we need to look at some example activity that could be used to evade the domain controller altogether.
Stealing local accounts is an excellent tool for lateral movement, so we want to walk through how responders might find indicators that a stolen account was used in the basic scenario of using a local administrator account at the User Account Control (UAC) prompt to run an application as administrator. There is absolutely no record of this activity on the domain controller, so let's check the host:

Code	Subject User Name	Subject Domain Name	Target User Name	Target Domain Name	Target Server Name	Target Info
4648	Alice	TESTDEV	Administrator	LABCLUB2-2	localhost	localhost
4624	Alice	TESTDEV	Administrator	LABCLUB2-2		
4672	Administrator	LABCLUB2-2				

Thanks to the host logs, we can get the context necessary to understand exactly what is occurring. An attacker with stolen credentials can no longer operate without the possibility for detection. It is just a matter of using the host event logs in the intelligent fashion originally set on the domain controller logs.

Pass-the-Hash (PtH) Using Metasploit

Next, let's look at the classic example of malicious account activity as a real test of host event logs. Using a relatively basic lab setup, we ripped the local administrator hash from one asset and successfully passed it to authenticate from a second host to a third.
Here is what appears in the Active Directory Security Logs on the domain controller:

Code	Target User Name	Target Domain Name	Workstation / Service Name / Logon Type	IP Address
4672	DC-01$	TESTDEV		
4624	DC-01$	TESTDEV	3 - Network	::1
4624	LABCLUB2-1$ (rip source)	TESTDEV	3 - Network	10.1.102.62 (rip source)

Not a whole lot to raise suspicions there, although the second 4624 might look like a ghost in the machine if we did not know that a PtH attack just occurred. We cannot alert every time a host appears on the domain controller to have successfully authenticated to itself and hope to find the rare PtH attack amid that noise. Now, look at what is shown on the target host:

Code	Subject User Name	Subject Domain Name	Target User Name	Target Domain Name	Workstation	IP Address	Logon Process Name
4672	Administrator	LABCLUB2-3					
4624			Administrator	LABCLUB2-3	**uxuQR742vgFacN18**	10.1.102.60 (source)	**NtLmSsp**

The new message code this time:

- 4672 - Special privileges assigned to new logon

This example clearly confirms that endpoint event logs contain evidence of PtH attacks, but how confidently can someone build alerting rules around this?

Mounting an Administrative Share - Domain Account

To learn more about these rules and their value, let us look at two examples of legitimate behavior that interact with the SMB protocol in similar ways to PtH. The first scenario is an administrative share being mounted through a domain account with local administrator permissions.
The relevant log messages on the domain controller for this activity show:

Code	Subject User Name	Workstation
4776	Alice	LABCLUB2-2

Once again, it is clear that *alice* attempted some type of authentication on *LABCLUB2*-2, but that authentication could be almost anything, save for the RDP example that we previously used because then there would be Kerberos ticket messages. We had better check out what the target host shows:

Code	Subject User Name	Subject Domain Name	Target User Name	Target Domain Name	Workstation	IP Address	Logon Process Name
4672	alice	TESTDEV					
4624			alice	TESTDEV	**LABCLUB2-2**	10.1.102.60 (source)	NtLmSsp

The target host log entries should look familiar because they are nearly identical messages for this activity and the PtH attack. Thankfully, the host event logs provide the account and source workstation, so that it is possible to distinguish this event from the previous.

Mounting an Administrative Share – Local Account

An ideal user base that would never use local administrator accounts to mount an administrative share, we would want an alert for both the PtH and the following example because each occurrence would mean that either a ripped hash is being passed or a stolen local account is in use.
Let's look at what appears on the host where *alice* instantiates this mount:

Code	Subject User Name	Subject Domain Name	Target User Name	Target Domain Name	Target Server Name
4648	alice	TESTDEV	Administrator	LABCLUB2-3	labclub2-3.testdev.com

We left out the domain controller logs for this activity because nothing of interest appeared. This is the case in almost any scenario where only local accounts are used, save for the random message that PtH generated on the host from which the hash was ripped. What is more interesting about comparing with PtH is that this log message never appeared on the source host because Metasploit directly implemented the SMB protocol outside of the Windows API.
With that in mind, we should look at what was logged on the target:

Code	Subject User Name	Subject Domain Name	Target User Name	Target Domain Name	Workstation	IP Address	Logon Process Name
4672	Administrator	LABCLUB2-3					
4624			Administrator	LABCLUB2-3	**LABCLUB2-2**	10.1.102.60 (source)	NtLmSsp

Well, the good news is that there are three differences between the sample Pass-the-Hash attack and this scenario where a local account is mounting an administrative share:

1. The bizarre 4624 message only appears on the domain controller when the hash is ripped.
2. The listed workstation is a legitimate host in both cases when the share is being mounted, but a random string in the Pass-the-Hash attack.
3. The *absence* of a 4648 message on the source host to link the authentication to the target host's 4624 message.

Realistically, though:

1. The 4624 might be challenging to use in incident response given the extremely high volume of similar messages logged every minute.
2. The best comparison point to distinguish PtH depends on the assumption that a real attacker can never spoof the workstation by defining the name of the process or some other means.
3. A great deal of false positives could arise from trying to alert on the lack of a correlated message on the source host, with causes ranging from interrupted log collection to faulty correlation, but this is also just a single example of Pass-the-Hash that happened not to use the Windows API.

Conclusion

This paper is not intended to end with readers feeling that detection is hopeless. Quite the contrary, we intend to demonstrate that by gathering the event logs from all hosts in an environment, compromised

accounts can be detected. This is not an exhaustive list of scenarios in which compromised accounts could be used and detected, but by touching on authentications as another domain account, a local account, and even a stolen hash, we enumerate just how much can be tracked across the various protocols in which a set of compromised credentials can be used.

Our conclusion after studying this data (and exploring a great many failed assertions) was that detecting the characteristics of stolen credentials in use requires context around typical behavior in an environment than simple rules-based alerting can provide. If focused exclusively on a PtH attack, one could likely detect it in use by a real attacker in most cases, but would miss all legitimate activity with which needs compared to other attacks in which a (phished or *mimikatz*-collected) password can be leveraged. To efficiently detect all uses of compromised credentials while minimizing false positives, one must gather all of this data and develop a method for learning expected ways in which accounts are used on each network. Raising the signal-to-noise ratio will require an understanding of both policy and practice, for example if *alice* commonly mounts administrative shares using local *Administrator* accounts and that is acceptable practice in an organization, responders need not be alerted on it each time.