

Anwendung des Least Privilege Principles bei
Service-Benutzern für Local Security Checks mit
Nessus

Christoph Pritz

7. Juni 2010

Abstract

In der heutigen Welt der Informationsgesellschaft ist der Computer ein Alltagsgegenstand. Millionen von Rechnern, vernetzt über das Internet ergeben neue interaktive Möglichkeiten um Information jedem zugänglich zu machen. Die Schattenseite dieser Entwicklung ist jedoch die steigende Kriminalität gegen Computer. Wie kann man in einer Welt, in der es Tausende von Hackern gibt, den Überblick über mögliche Schwachstellen der zu schützenden Rechner bewahren? Die Lösung sind automatische Penetration Tools wie zum Beispiel Nessus. Nessus scannt gegebene Rechner im Netzwerk und listet deren Schwächen auf, damit eine Kompromittierung des Systems erschwert wird. Nessus kann dabei das System nicht nur über das Netzwerk scannen (Port Scanning, etc.) sondern auch mittels gegebener Authentifizierungsdaten einen Local Check durchführen, der viele Vorteile besitzt. In der Unix Welt geschieht dies mittels SSH, in der Windows Welt mit Hilfe von SMB. Dabei sollte der Benutzer möglichst wenig Rechte haben, um Nessus nur die nötigsten Informationen zu geben und im Falle einer Kompromittierung das Schadensausmaß so gering wie möglich zu halten. In dieser Arbeit wurden die Informationen bzw. Daten recherchiert, die Nessus benötigt, um das System einschätzen zu können. Der Zugriff kann danach auf Windows und Unix so eingeschränkt werden, dass Nessus nur mehr Zugriff auf diese relevanten Datenbestände hat.

Inhaltsverzeichnis

1	Einleitung	4
1.1	Local Security Checks	4
2	Linux	6
2.1	Nessus Konfiguration	6
2.2	”Filemon” unter Linux	8
2.3	Implementierung	9
2.3.1	Andere Möglichkeiten	10
3	Windows	11
3.1	Erlauben des SMB Zugriffs	11
3.2	Monitoring unter Windows	11
3.3	Nessus Konfiguration	13
3.4	Ergebnisse	13
3.5	Implementierung	14
4	Zusammenfassung	15
5	Quellen	15

1 Einleitung

Nessus wurde von Tenable Network Security entwickelt und ist ein Penetration-Testing Tool basierend auf der Client-Server Architektur. Der Server beinhaltet zentral alle Plugins, die für das Angreifen der Systeme erforderlich sind. Die Plugins werden vom Hersteller für Privatpersonen gratis, oder für professionelle Anwender im Firmenumfeld gegen eine jährliche Gebühr zur Verfügung gestellt. Nessus scannt primär netzwerkbasierend, d.h. es wird zuerst ermittelt um welchen Typ von Host es sich handelt und welche Netzwerkdienste darauf laufen, danach wird in der Datenbank des Servers eruiert ob es für diesen Netzwerkdienst eine Schwäche gibt.

1.1 Local Security Checks

Eine andere Methode die Nessus anbietet ist der sogenannte Local Security Check. Dabei loggt sich Nessus mit vorher bekanntgegebenen Zugriffsdaten im Remotesystem ein und überprüft dieses, was folgende Vorteile bringt (vgl. Tenable Network Security o.J., o.S.):

+ *Erkennen von Schwächen in Clientprogrammen:*

Viele Schwächen von Clients können nicht über das Netzwerk erkannt werden. Ein Local Security Check prüft, ob das Programm, bzw. das Betriebssystem Sicherheitslücken aufweist. Nessus kann optimalerweise sogar den Patch vorweisen, der benötigt wird um die Lücke zu schließen.

+ *Erkennen von nicht aktiven Services:*

Falls ein Service gerade nicht in der Zeit läuft, in der der Netzwerkan aktiv ist wird dieser Service logischerweise nicht erkannt und daher auch nicht auf Schwächen überprüft. Der Local Security Check kann diese Dienste erkennen und auf Schwachstellen untersuchen.

+ *Geschwindigkeit:*

Ein Local Security Check verbraucht nur die Bruchteile der Zeit eines normalen Network Checks, da man sich nicht mit Portscanning und anderen zeitaufwändigen Techniken beschäftigen muss, um die Services zu erkennen die auf dem Client laufen.

+ *Weniger Eingriff in den Betrieb:*

Ein Network Check verbindet sich auf vielen TCP Ports mit dem Zielsystem, was eine Belastung für dieses zur Folge hat. Dies kann im schlimmsten Fall zu einer Beeinträchtigung des Netzwerkbetriebs führen. Ein Local Check verbindet sich nur per Remote Shell mit dem System, was dieses wenig bis gar nicht belastet.

+ *Genauigkeit:*

In manchen Fällen ist es nicht möglich über das Netzwerk zu prüfen, ob ein Service eine Schwachstelle aufweist. Ein Local Security Check kann die Anzahl dieser "False Positives" verringern.

Momentan werden alle gängigen Betriebssysteme unterstützt. Was jedoch passiert wenn der Benutzer, mit dem sich Nessus am Remotesystem anmeldet, kompromittiert wird? Weiters soll sichergestellt werden, dass Nessus nur die Daten begutachtet, die der Scanner auch wirklich braucht um ein Urteil über den Zustand des Systems zu fällen. Daher wird in dieser Arbeit der Ansatz verfolgt dem Benutzer am Remotesystem, dem sogenannten Service User, das Principle of Least Privilege aufzuzwingen. Dieses besagt, dass jeder Nutzer nur so viele Rechte als nötig haben sollte, um bei einer Kompromittierung das Schadensausmaß so gering als möglich zu halten. Als Testobjekte werden in dieser Arbeit die zwei gängigen Betriebssystem Windows XP und CentOS behandelt.

Forschungsfrage

Wie kann das Least Privilege Principle bei einem Service Benutzer für Local Security Checks mit Nessus in den Betriebssystemen Windows XP und Linux implementiert werden?

2 Linux

Für den Übungsversuch in diesem Dokument wird die CentOS Version 5.4 64 Bit verwendet. Dieses wurde in einer VM-Ware aufgesetzt. Nach dem Aufsetzen kann bereits der Service User (nessus) und die dazugehörige Gruppe(nessusgroup) erstellt werden, der danach für die Local Security Checks verwendet wird:

```
groupadd nessusgroup
useradd -c "Nessus Service User" -d /home/nessus -m
-g nessusgroup -s /bin/bash nessus
```

Nun kann bereits der SSH-Zugang eingerichtet werden, indem mit dem Tool "puttygen.exe" ein RSA Schlüsselpaar generiert wurde. Der Public Key muss danach auf alle zu scannenden Rechner in das "authorized_keys" im Home Directory des Service Benutzers übertragen werden. Schlussendlich müssen noch die entsprechenden Rechte gesetzt werden.

```
chown -R nessus:nessusgroup ~nessus/.ssh/
chmod 0600 ~nessus/.ssh/authorized_keys
chmod 0700 ~nessus/.ssh/
```

Danach ist ein Login über SSH ohne Username und Passwort mit dem Benutzer bereits möglich (siehe Abbildung 1). Dies ist für einen funktionierenden Local Check unbedingt erforderlich. Anmerkung: Nessus kann sich für den Local Check auch mit vorher angegebenen Authentifizierungsdaten (z.B.: Username + Passwort) authentifizieren. In unserem Fall wird jedoch Public Key Kryptografie verwendet, da es mehr Sicherheit und Komfort bietet.

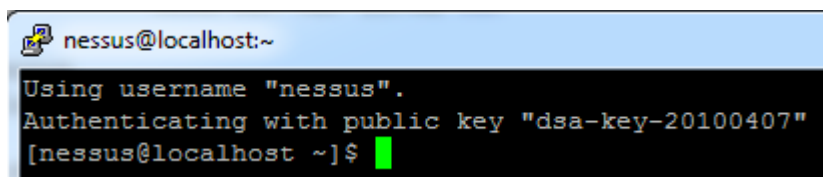


Abbildung 1: Funktionierender Login mit dem Service Benutzer per SSH

2.1 Nessus Konfiguration

Um Nessus zu konfigurieren muss der Nessus Client aufgerufen werden. Dieser ist in den neueren Versionen ein Flash Interface, das im Browser auf-

gerufen wird. Dort muss danach eine neue Policy erstellt werden, in der dem Client über das Menü "Credentials" mitgeteilt wird, welche Authentizitätsinformationen benutzt werden (vgl. Tenable Network Security 2009, S. 12). In diesem Fall wird dem Client das asymmetrische Schlüsselpaar mitgeteilt, mit dem er die SSH Kommunikation vornehmen kann (siehe Abbildung 2).

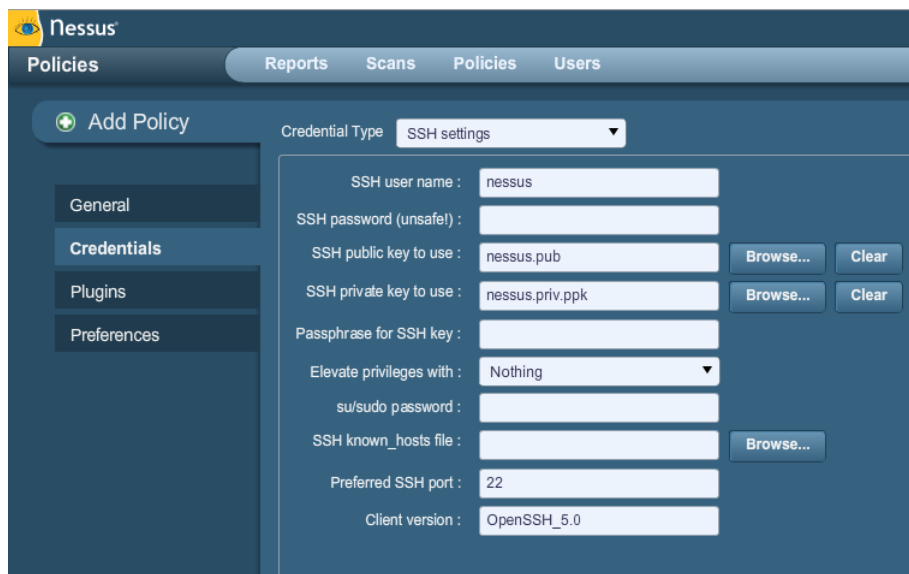


Abbildung 2: Bekanntgabe der SSH Authentifizierungsdaten

Schlussendlich muss noch ein Scan definiert werden. Dort wird die vorher definierte Scan Policy ausgewählt und die IP-Adresse des CentOS Linux eingetragen. Danach kann bereits der Scan gestartet werden, um die Funktion des Local Security Checks zu überprüfen. Ob der Local Security Check funktioniert hat, sieht man danach in den Log-Dateien von Nessus (siehe Abbildung 3).

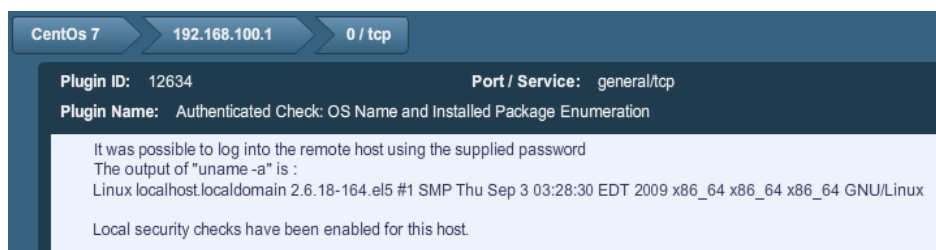


Abbildung 3: Der Local Security Check wurde erfolgreich durchgeführt

2.2 "Filemon" unter Linux

Das Programm Filemon ermöglicht in Windows das Betrachten der gerade offenen Files eines Prozesses. Dies ist notwendig, um zu sehen welche Files von Nessus angesehen werden, damit diese ein Urteil fällen kann. Für diesen Zweck musste ein Filemon auf Linux mit Hilfe von gängigen Befehlen nachgebaut werden. Dies wurde in einem Shellskript realisiert:

```
touch file_access
ps aux | grep nessus > l1

while [ 1 ]
do
    ps aux | grep nessus > l2
    diff l1 l2 >> file_access
    rm l1
    mv l2 l1
done

grep -v root file_access > file_access_filtered
```

Listing 1: Der Quelltext des "filemon" Skripts

Die Funktions ist simpel erklärt: Das Skript listet alle laufenden Prozesse des Benutzers Nessus auf und schaut immer wieder ob neue Prozesse hinzugekommen sind (Anmerkung: Nachträglich wurde herausgefunden, dass der Vergleich auch mit dem "watch" Kommando erfolgen könnte). Dies muss natürlich in einer Schleife erfolgen. Zum Schluss wird die Ausgabe noch gefiltert, da auch der "grep" Befehl vom User "root" aufscheint.

Mit diesem Skript am Laufen wurde ein Local Security Check durchgeführt. Dabei führt Nessus laut dem Skript folgende Aktionen durch:

- **"uname -a"**
Bestimmung der Kernelversion & -architektur, Domäne und eingestelltes Datum
- **"rpmq -q -all -qf %NAME-%VERSION-%RELEASE-%EPOCH"**
Bestimmen der aktuell installierten Packages mit dazugehöriger Version bzw. Release

- **”dmidecode”**
Ermitteln der Hardwarekomponenten, Seriennummern und Biosrevision.
- **”netstat -anp”**
Dieses Kommando zeigt die offenen Netzwerkverbindungen an.
- **”cat /etc/passwd”**
Nessus listet mit diesem Kommando alle möglichen Benutzer dieses Computers auf.
- **Überprüfen der SSH Trusted Hosts Dateien**
Nessus begutachtet im Home Directory jedes Benutzers die Dateien für die SSH Trusted Host Verbindung. Was genau Nessus damit bezweckt kann nur angenommen werden. Es wird jedoch vermutet, dass Nessus die exklusiven Leserechte der jeweiligen Benutzer auf die Dateien verifiziert.

2.3 Implementierung

Auf Linux hat man das Problem einem Nutzer nur bestimmte Befehle mit bestimmten Parametern zu gewähren. Mit ”sudo” ist zwar eine Einschränkung auf Befehle möglich, jedoch ist die Einschränkung bei den Parametern nur begrenzt möglich. Eine Lösung für das Problem würde jedoch ein sogenanntes ”Wrapper Script” darstellen. Zuerst sollte man auf dieser Maschine bei jeder anderen Datei die ”rwx” Bits bei ”Others” entfernen. Dies verbietet allen Usern, die nicht in einer bestimmten Gruppe sind den Zugang zu den Dateien bzw. Programmen. Somit kann der Service User prinzipiell nichts ansehen oder ausführen. Der Clou ist jetzt der Einsatz der Wrapper Scripts, welche prinzipiell wie eine eingeschränkte Shell wirken. Diese Scripts werden wie die Befehle genannt und werden in einem Ordner abgelegt, der danach in der \$PATH Variable hinzugefügt wird (über das ”.profile” Skript). Dieses Skript führt den gleichen Befehl wie den Skriptnamen aus, jedoch werden vorher die Parameter überprüft. Diese Skript besitzt das SUID Bit, was das Skript in der Ausführung auf Root Ebene hebt, und ist vom Service Benutzer nicht schreibbar. Somit kann der Service Benutzer nur die Befehle ausführen, die ihm vorgegeben werden.

Folgendes Skript würde die Ausführung von "netstat -anp" genehmigen:

```
if [ "$1" = "-anp" ]
then
    /bin/netstat -anp
else
    echo "Keine_Berechtigung"
fi
```

Listing 2: Ein Beispiel für das "netstat" Wrapper Skript

Würde man für jeden Befehl ein eigenes Skript erstellen, sollte die Einschränkung durchaus funktionieren.

2.3.1 Andere Möglichkeiten

Es wurde ebenfalls überlegt die Restriktion mittels ACLs umzusetzen. Das Problem ist, dass hier keine Überprüfung auf Parameter erfolgen kann. Zusätzlich dazu kam die Überlegung, ob eine ACL auf eine oft genutzte Datei nicht die Systemperformance, wenn auch nur im kleinen Maße, einschränkt.

3 Windows

Für diese Aufgabe wurde ein Windows XP SP2 aufgesetzt und in ein Netzwerk mit dem Nessus Server eingebunden. Die Firewall wurde deaktiviert.

3.1 Erlauben des SMB Zugriffs

Damit der Nessus Server auf den Client mittels SMB zugreifen kann, muss zuerst ein bestimmter Parameter gesetzt werden. Falls sich der Computer in einer Domäne befinden würde wäre dies nicht notwendig. Der Eintrag findet sich in der Systemsteuerung → Verwaltung → Lokale Sicherheitseinstellungen → Lokale Richtlinien → Sicherheitsoptionen. Hier ist der Eintrag "Netzwerkzugriff: Modell für gemeinsame Nutzung und Sicherheitsmodell für lokale Konten" auf "Klassisch - lokale Benutzer authentifizieren sich als sie selbst" zu ändern, damit der Zugriff funktioniert (siehe Abbildung 4).

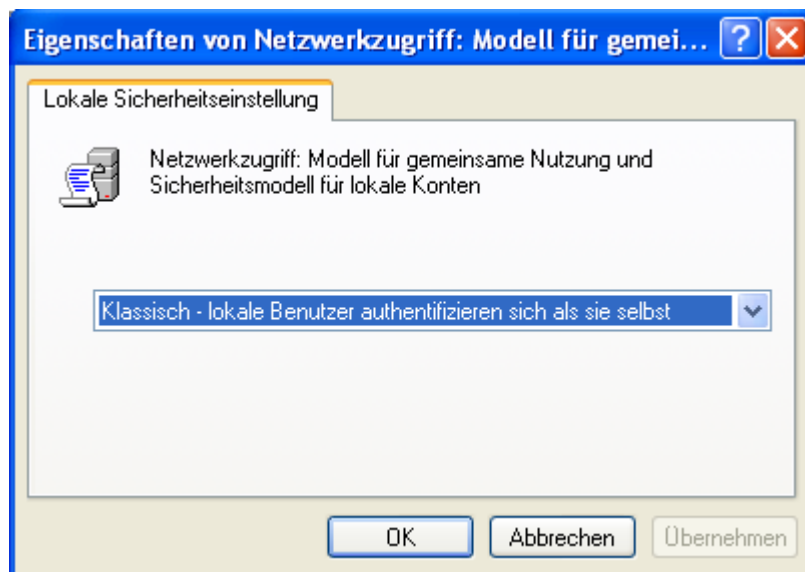


Abbildung 4: Erlauben des SMB Zugriffs

3.2 Monitoring unter Windows

Im Gegensatz zu Unix gibt es auf Windows bereits viele Monitoring Tools. Microsoft hat dabei einen sehr praktische Programm kreiert, dass alle laufenden Prozessaktionen und Registryzugriffe aufzeichnet. Dieses Programm nennt sich Procmon und ist unter "<http://technet.microsoft.com/en-us/>

sysinternals/bb896645.aspx” zum Download verfügbar. Damit unwich-

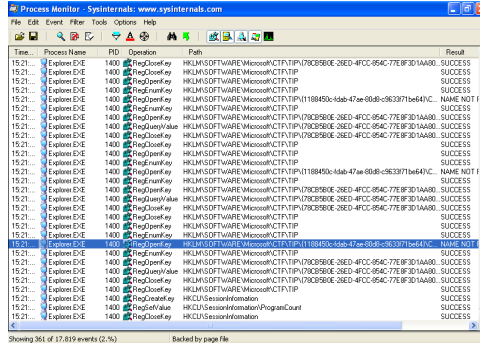


Abbildung 5: Das Programm Procmon im Einsatz

tiger Output die Analyse nicht stört, wurden ein paar Filterregeln erstellt, die das Anzeigen von oft ausgeführten Befehlen verhindern.

WMI

Damit die WMI Befehle die Nessus ausführt sichtbar sind, muss das Logging Level erhöht werden. Dies geschieht unter Computerverwaltung → Dienste und Anwendungen → RM auf WMI-Steuerung → Eigenschaften → Protokollierung. In dieser Kartei muss die Auswahl ”Ausführlich (enthält Zusatzinformationen für den Microsoft Ratgeber)” (siehe Abbildung 6) ausgewählt werden (vgl. CC Hameed 2008). Danach ist das Logging Level sofort ohne einen Neustart erhöht.

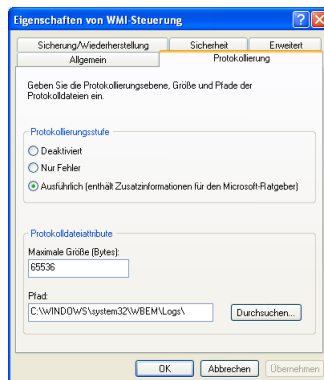


Abbildung 6: Erhöhen des WMI Logging Levels

3.3 Nessus Konfiguration

Genauso wie beim Einstellen unter Unix muss hier auch wieder eine neue Policy definiert werden, wobei hier aber SMB Credentials vergeben werden müssen (vgl. Tenable Network Security 2009, S. 16)(siehe Abbildung 7). Danach muss erneut diese Policy in einem Scan verpackt mit der IP-Adresse des Windows Hosts konfiguriert werden.

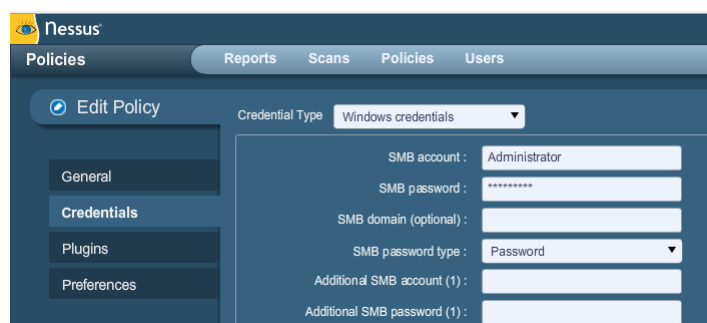


Abbildung 7: Konfigurieren der SMB Authentifizierungsdaten

3.4 Ergebnisse

Mit den laufenden Monitoring Tools wurde nun ein Scan durchgeführt. Aufgrund der ca. 50.000 Registry Abfragen können die abgefragten Einträge nicht einzeln angeführt sondern im großen Rahmen eingegrenzt werden. Folgende Abfragen wurden getätigt

- **HKEY_CLASSES_ROOT:** Alles
- **HKEY_CURRENT_USER:** \Console, \Control Panel, \Environment, \Session Information, \Software
- **HKEY_LOCAL_MACHINE:** \HARDWARE, \SAM\SAM, \SOFTWARE, \SYSTEM\CurrentControlSet\Enum, \SYSTEM\CurrentControlSet\Services, \SYSTEM\CurrentControlSet\Control, \SYSTEM\Setup, \SYSTEM\WPA
- **HKEY_USERS:** \.DEFAULT\Control Panel, \.DEFAULT\Software, \S-1-5-20 (Das ist die SID des Network ServiceAccounts!)

WMI Calls gab es keine Erwähnenswertes. Der Local Check basiert zu 99,9% aus Registry Calls.

3.5 Implementierung

Nessus braucht auch auf Windows Maschinen keinen Administrator Account. Es würde genügen nur die oben erwähnten Registry Pfade für den Service Benutzer zugänglich zu machen. Dies geschieht im Registry Editor "regedit". Dort kann man mit Rechtsklick → Berechtigungen, ähnlich wie bei normalen Dateien, die Rechte für diesen Teil der Registry vergeben (siehe Abbildung 8). Genauso wie beim normalen NTFS Berechtigungsmodell ist hier auch Vererbung möglich.

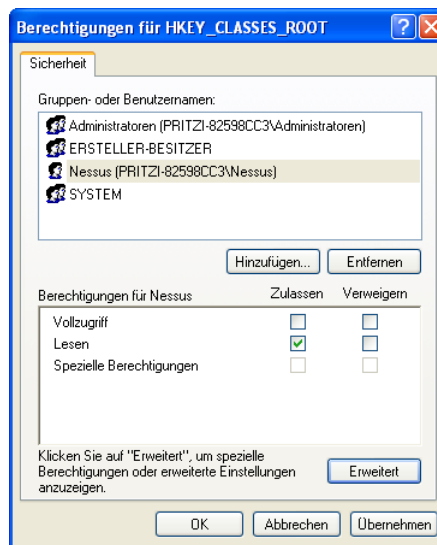


Abbildung 8: Berechtigungen für bestimmte Registry Pfade setzen

4 Zusammenfassung

Die Local Checks von Nessus benötigen auf keinen Fall zwingend Administratorrechte, um alle Schwächen zu entdecken. In Unix führt der Local Check eine Hand voll Befehle aus, die zum Beispiel durch ein Wrapper Script im ".profile" Ordner die einzigen Befehle sein könnten, die der Local Check effektiv ausführen darf. Auf der Windows Ebene werden viele Registry Calls abgesetzt, die durch Berechtigungen auf die notwendigen Registrypfade limitiert werden können. Nachdem diese Restriktionen implementiert wurden, waren die Ergebnisse, die Nessus lieferte, noch immer ident.

Aufbauend auf diese Arbeit könnte zum Beispiel ein Vergleich bzw. eine Verbesserung verschiedener praktischer Implementierung dieser Konzepte untersucht werden. Dabei können die Erkenntnisse dieser Arbeit, nämlich die Daten auf die Nessus zugreift, stark von Vorteil sein.

5 Quellen

CC Hameed. (2008). WMI Debug Logging.

[<http://blogs.technet.com/askperf/archive/2008/03/04/wmi-debug-logging.aspx> (ausgehoben 19.05.2010)]

Tenable Network Security. (2009). Nessus Credential Checks.

[http://www.nessus.org/documentation/nessus_credential_checks.pdf (ausgehoben 23.04.2010)]

Tenable Network Security. (o.J.). The theory behind local security checks.

[<http://www.nessus.org/documentation/index.php?doc=ssh> (ausgehoben 12.04.2010)]

Abbildungsverzeichnis

1	Funktionierender Login mit dem Service Benutzer per SSH . . .	6
2	Bekanntgabe der SSH Authentifizierungsdaten	7
3	Der Local Security Check wurde erfolgreich durchgeführt . . .	7
4	Erlauben des SMB Zugriffs	11
5	Das Programm Procmon im Einsatz	12

6	Erhöhen des WMI Logging Levels	12
7	Konfigurieren der SMB Authentifizierungsdaten	13
8	Berechtigungen für bestimmte Registry Pfade setzen	14