BRAIN-MEDIA.DE

PROFESSIONELLES COMPUTER- UND INTERNET-KNOW-HOW

Holger Reibold

OpenVAS kompakt

Alles, was Sie für den erfolgreichen Einsatz des freien Security-Scanners wissen müssen

Holger Reibold

OpenVAS kompakt

© 2010 Brain-Media.de

Herausgeber: Dr. Holger Reibold

Umschlaggestaltung: Brain-Media.de

Satz: Brain-Media.de

Coverbild: Photocase/froodmat

Druck: COD

ISBN: 978-3-939316-73-2

Inhaltsverzeichnis

Vorwort

In den letzten Jahren sind Bedrohungen in der Informationstechnik und die Zahl von Angriffen gegen IT-Systeme ständig gewachsen. Bei gleichzeitig zunehmender Bedeutung der Informations- und Kommunikationstechnologie (IKT) in der heutigen Gesellschaft ist es daher erforderlich, das Sicherheitsniveau der Informationstechnik ständig zu verbessern. Dies ist eine Aufgabe des Bundesamts für Sicherheit in der Informationstechnik (BSI).

Um die Sicherheitseigenschaften von Software nachweisen zu können, muss sie prüfbar sein. Wesentliche IT-Werkzeuge erstellt bzw. fördert das BSI daher als Freie Software. Ob bei Servern, Netzwerkgeräten, Routern, Netzwerkdruckern oder Einzelplatzrechnern – Sicherheitslücken müssen rechtzeitig aufgespürt und geschlossen werden, damit man sich vor Angreifern schützen kann.

Ein Open-Source-Software-Tool, das Schwachstellen identifiziert und Hilfestellungen für die Beseitigung der Lücken gibt, ist das Open Vulnerability Assessment System (OpenVAS), die konsequente freie Weiterentwicklung des seit 2005 proprietären Security-Scanners Nessus. Neben der Weiterentwicklung und Erweiterung des Security-Scanners hat das BSI auch die freie Entwicklung von Prüfmodulen in Kooperation mit der Wirtschaft gefördert.

OpenVAS ist im Laufe der Jahre zu einem ansehnlichen und vertrauenswürdigen Sicherheitswerkzeug für den Bürger herangewachsen. Einer Studie zufolge liegt OpenVAS bei aktuellen Sicherheitsprüfungen vorn. Aus diesem Grund ist es bereits in aktuellen Linux-Distributionen verfügbar.

Das BSI legt Wert auf eine einfache Bedienbarkeit, um möglichst vielen Bürgern den Zugang zu Open-Source-Sicherheitsprodukten zu ermöglichen. Eine einfache und intuitive Handhabung war daher immer im Blick der OpenVAS-Entwickler. Dennoch ist es für Einsteiger wichtig, weitere Hilfen zur richtigen und effizienten Nutzung des Produkts erhalten zu können. Mit dem vorliegenden Buch sollen diese dem Nutzer an die Hand gegeben werden.

Sichere Stunden in der virtuellen Welt wünscht

Ihr Michael Hange

Präsident des BSI

Einführung

Sicherheitsprodukte haben Hochkonjunktur. Kein Wunder, denn fast täglich kann man der Presse Meldungen über kritische Sicherheitslücken, erfolgreich gehackte Server, Datenklau und Manipulationen entnehmen. So steigt die Nachfrage nach Werkzeugen, die helfen, Sicherheitslücken aufzuspüren und zu schließen. OpenVAS ist ein solches Tool, das zur Kategorie der Vulnerability-Scanner gehört. Solche Scanner nehmen beliebige Netzwerkgeräte, Server, Netzwerkdrucker, Router und natürlich auch Einzelplatzrechner unter die Lupe, identifizieren Schwachstellen und geben Hilfestellung beim Schließen von Lücken.

In der Welt der Sicherheitsscanner galt Nessus (*http://www.nessus.com*) lange als der Standard schlechthin. Doch mit der Einführung von Nessus 3.0 wurde aus dem einst freien Projekt eine propritäre Lösung. Dem Intevation-Team (*http://intevation.de*) haben wir es zu verdanken, dass es einen freien Nachfolger gibt: OpenVAS (*http://www.openvas.org*). Dieses Tool unterliegt nicht nur der GPL, sondern wird auch maßgeblich von deutschen Entwicklern weiter ausgearbeitet. Auch das Bundesamt für Sicherheit in der Informationstechnik ist an der Weiterentwicklung beteiligt. Mit der Einführung von OpenVAS 3.0 hat der Vulnerability-Scanner auch einige technologische Neuerungen und Verbesserungen zu bieten, dank denen OpenVAS seinem Vorläufer den Rang abläuft.

Einziges Manko von OpenVAS: die unzureichende Dokumentation. Das vorliegende Buch soll helfen, diese Lücke zu schließen. In **OpenVAS kompakt** erfahren Sie, wie Sie den Security-Scanner einsetzen, wie Sie Tests durchführen und Berichte interpretieren. Sie erfahren außerdem, wie Sie eigene Test-Skripts erzeugen. Tipps und Tricks für den Praxiseinsatz sowie interessante Hintergrundinformationen runden das Buch ab.

Das Buch ist mit tatkräftiger Unterstützung des OpenVAS-Teams entstanden. An dieser Stelle möchte ich stellvertretend für viele andere Dr. Jan-Oliver Wagner von Intevation danken. Mein besonderer Dank gilt Michael Meyer von Greenbone Networks (*http://www.greenbone.net*), der jede noch so „dumme“ Fragen mit Engelsgeduld beantwortet hat. Ohne seine und die Hilfe anderer wäre das Buch niemals so fundiert geworden, wie es ist.

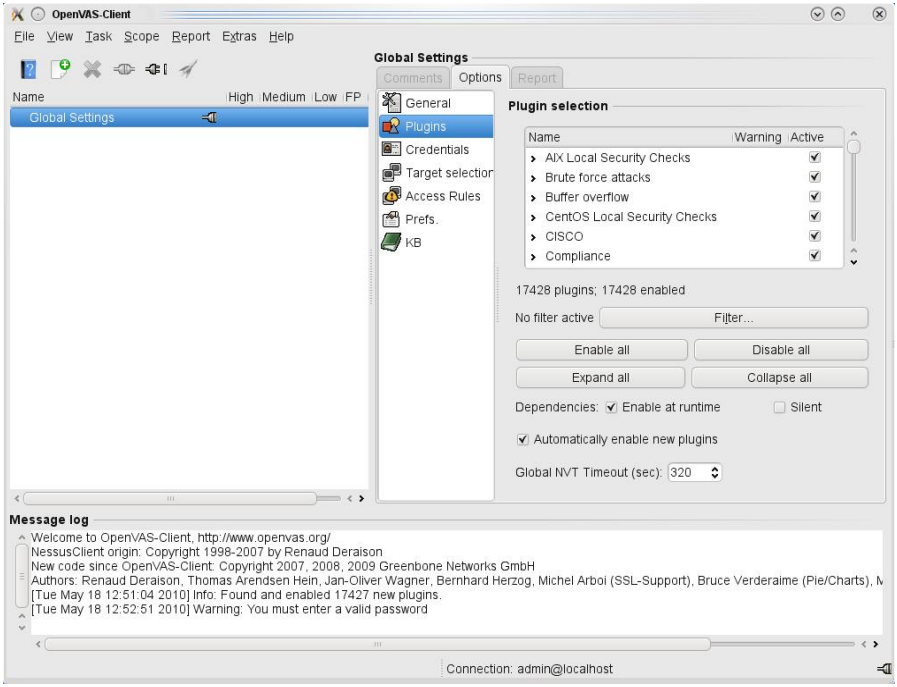
Viel Erfolg beim Aufspüren und Schließen von Verwundbarkeiten wünscht

Holger Reibold (Mai 2010)

1 OpenVAS − der Einstieg

Wenn Sie Ihr Netzwerk und Ihre kritischen Systeme auf ihre Zuverlässigkeit hin überprüfen wollen, müssen Sie diese wohl oder übel echten Attacken aussetzen.

Da Sie sich oder einen Ihrer Mitarbeiter nicht erst zum Hacker ausbilden wollen und können, sollten Sie zu einem Spezialisten greifen, der all das beherrscht. Die Rede ist von einem Security-Scanner. Solche Systeme setzen die zu testenden Systeme typischen Attacken aus – oder simulieren dies zumindest.

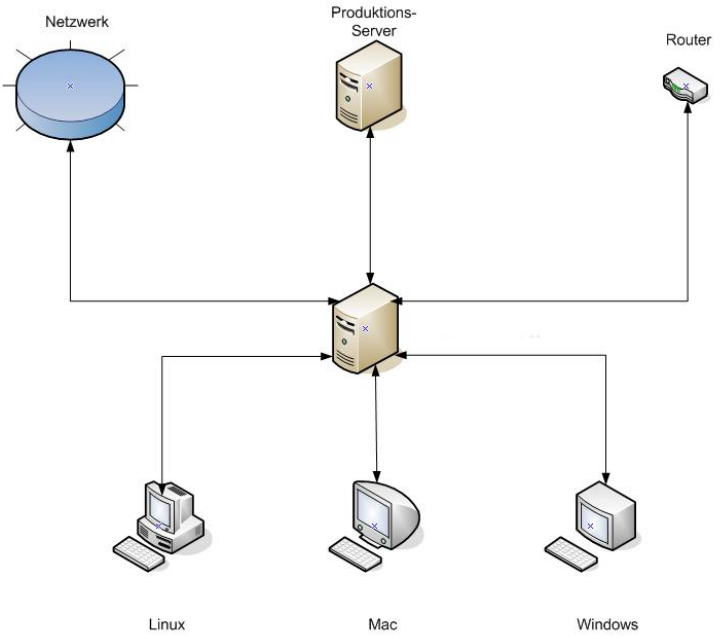


Ein erster Blick auf den OpenVAS-Client – genauer auf Version 3.0.0 –, mit dem Sie den OpenVAS-Server steuern.

Der einzige freie Sicherheitsscanner, der auch professionellen Anforderungen genügt, ist OpenVAS. Dieses Tool ist ein typischer Vertreter der Netzwerk- oder Vulnerability-Scanner.

Das Tool basiert auf einer Client-Server-Architektur. Der Server wird auf einem Linux-System (openvas-scanner) ausgeführt und kann von einem lokalen oder auch entfernten Client gesteuert werden.

Beim Start des Servers werden automatisch sogenannte Plug-ins geladen, mit denen sich diverse Sicherheitslücken des Betriebssystems bzw. der Dienste aufdecken lassen, die auf den zu scannenden Hosts laufen. Die Plug-ins werden in der OpenVAS- bzw. Nessus-eigenen Skriptsprache Nessus Attack Scripting Language (NASL) erstellt.



Die typische OpenVAS-Architektur: Die Clients, insbesondere auf Linux- oder Windows-Rechnern, steuern den zentralen OpenVAS-Server, der die Sicherheitstests auf beliebigen Systemen ausführt.

Das Prinzip ist ansonsten recht einfach: Der Client stellt eine Verbindung zu dem Server her, erzeugt eine Session, in der die Plug-ins, der oder die Ziel-Hosts und weitere Scan-Einstellungen definiert werden.

Wurde der Scan auf einem Host ausgeführt, gibt der OpenVAS-Client eine Übersicht über die offenen Ports und eventuell gefundene Sicherheitslücken aus.

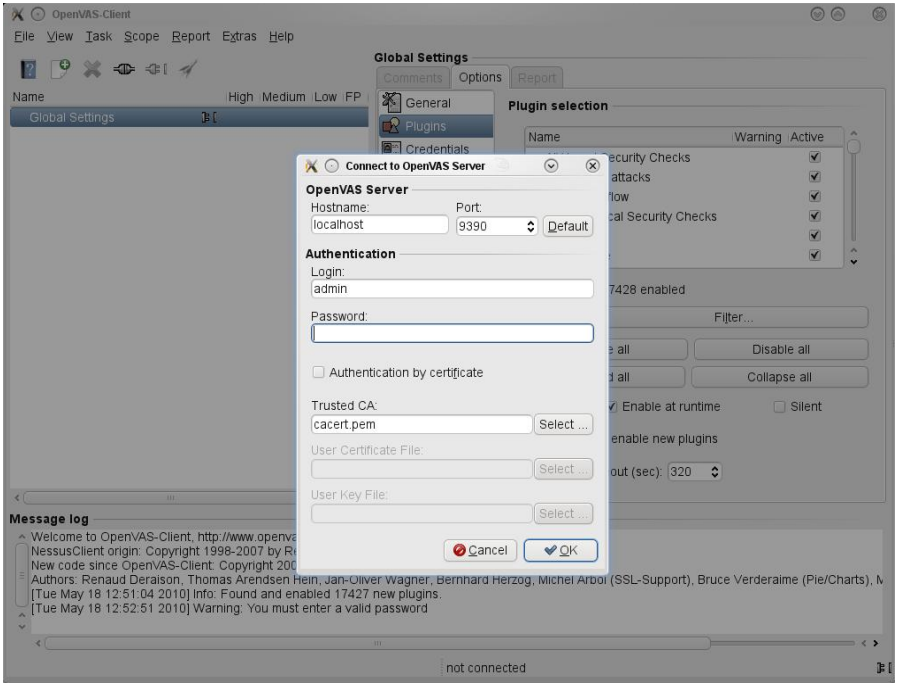
1.1 OpenVAS-Quickstart

Im Zusammenhang mit Security-Scannern taucht auch immer wieder der Begriff des Penetrationstests auf. Ein Penetrationstest sollte jedoch ein zielgerichteter Angriff mit den Mitteln eines Angreifers sein. Angreifer verwenden keine Security-Scanner für ihren Angriff, da diese Programme zu massiv Eintragungen in Logfiles bzw. Intrusion-Detection-Systemen hinterlassen. Ein Penetrationstest ist also der Versuch, mit den Mitteln eines Angreifers und innerhalb einer gegebenen Zeitspanne Lücken in der IT-Sicherheit aufzudecken. Auch hierfür gibt es Hilfsmittel.

Beim Einsatz eines Security-Scanners sind folgende Punkte zu beachten:

- Portscans können ein System lahm legen. Daher sollte bei unbekannter Hard- oder Software zuerst ein Test auf Nicht-Produktionssysteme stattfinden.
- Kritisch ist der Scan von Produktionssystemen, da ein Ausfall des Systems aufgrund eines Scans nie ganz ausgeschlossen werden kann.
- Besonders kritisch ist eine Überprüfung von relevanten Systemen.

Unternehmen wir die ersten Schritte. Ist OpenVAS installiert und der OpenVAS-Server gestartet, können Sie mit dem OpenVAS-Client eine Verbindung zum Server aufbauen. Dazu starten Sie den OpenVAS-Client. Verschaffen wir uns einen ersten Überblick. Der Client präsentiert Ihnen einen einfachen Dialog, in dem die zu scannenden Systeme bzw. Netzwerke und die Scan-Richtlinien aufgeführt werden.

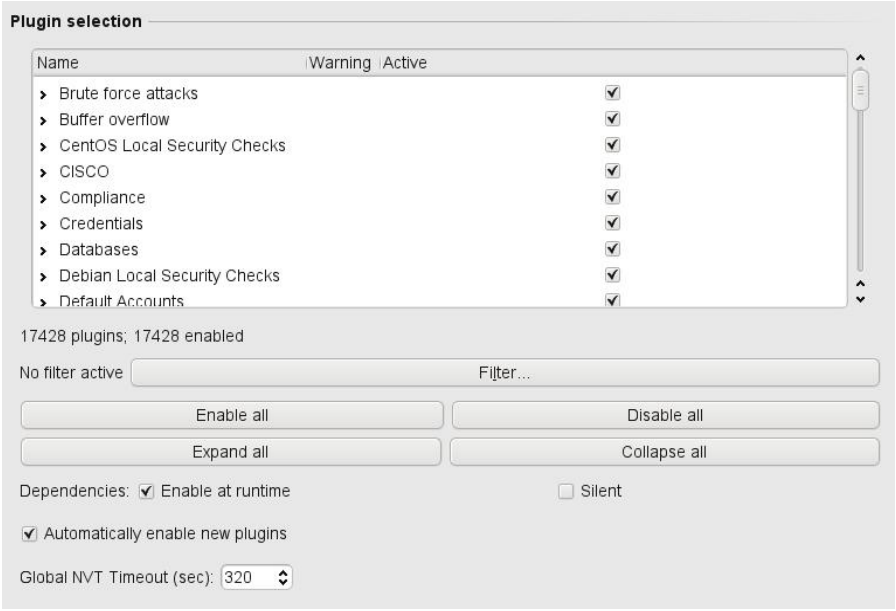


Die Einstellungen für den Verbindungsaufbau zum lokalen OpenVAS-Scanner.

Prinzipiell können Sie mit dem Client mehrere OpenVAS-Server steuern.

Im *Connect to OpenVAS Server*-Dialog finden Sie bereits den Eintrag *localhost*, der die Nutzung eines lokal installierten Servers vorsieht. In den Verbindungseinstellungen geben Sie den Hostnamen, den Standardport des Servers (in der Regel 9390) und die Zugangskennung an.

Wenn Sie OpenVAS-Scanner und -Client auf verschiedenen Systemen einsetzen, sollten Sie außerdem die SSL-Einstellungen bearbeiten, um die Verbindung zwischen beiden zu sichern. Sie stellen die Verbindung mit einem Klick auf *OK* her.



Die Plug-in-Auswahl.

Als Nächstes sind die Scan-Einstellungen dran. Auch hier kommt der Client mit einem vordefinierten Standardprofil daher. Sie können weitere Scan-Profile anlegen und diese nach Belieben bearbeiten.

Achtung

Da die Plug-ins vom Server, nicht aber vom Client bereitgestellt werden, ist die Plug-in-Liste nur dann verfügbar, wenn Sie eine Verbindung zu einem OpenVAS-Server aufgebaut haben.

1.2 Das Herzstück: die Plug-ins

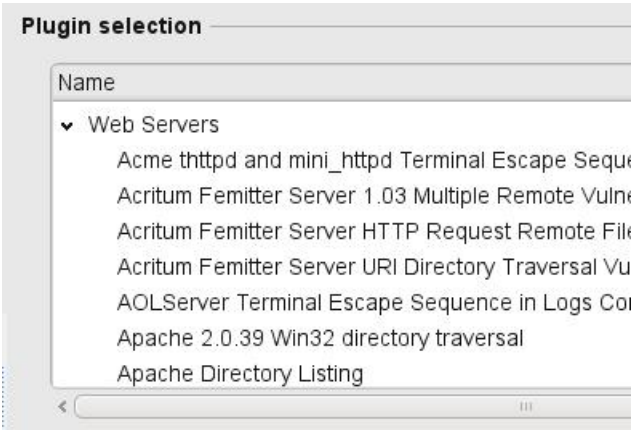
OpenVAS präsentiert Ihnen vier Registerkarten. Auf dem Register *Plugin Selection* finden Sie die Einstellungen für die Plug-ins. Wie bereits erwähnt, handelt es sich bei den Plug-ins um das eigentliche Herzstück des Scanners. Hinter den Plug-ins verbergen sich die Sicherheitstests, die einen möglichen Angriffspunkt entde-

cken. In der OpenVAS-Terminologie ist der Begriffs, der Plug-ins am Verschwinden. Die Begriffe Plug-ins und NVTs sind gleichbedeutend.

Im Mai 2010 gab es über 40 Plug-in-Kategorien mit mehr als 17.000 Plug-ins. Die Kategorien:

- AIX Local Security Checks
- Brute force attacks
- Buffer overflow
- CISCO
- CentOS local Security Checks
- Databases
- Debian Local Security Checks
- Default Unix Accounts
- Denial of Service
- FTP
- Fedora Local Security Checks
- Finger abuses
- Firewalls
- FreeBSD local Security Checks
- Gain a shell remotely
- Gain root remotely
- General
- Gentoo Local Security Checks
- HP-UX Local Security Checks
- IT Grundschutz
- IT Grundschutz-10
- IT Grundschutz-11
- Mac OS X Local Security Checks
- Malware

- Mandrake Local Security Checks
- Netware
- NIS
- Peer-To-Peer File Sharing
- Privilege escalation
- Red Hat Local Security Checks
- Remote file access
- RPC
- Service Detection



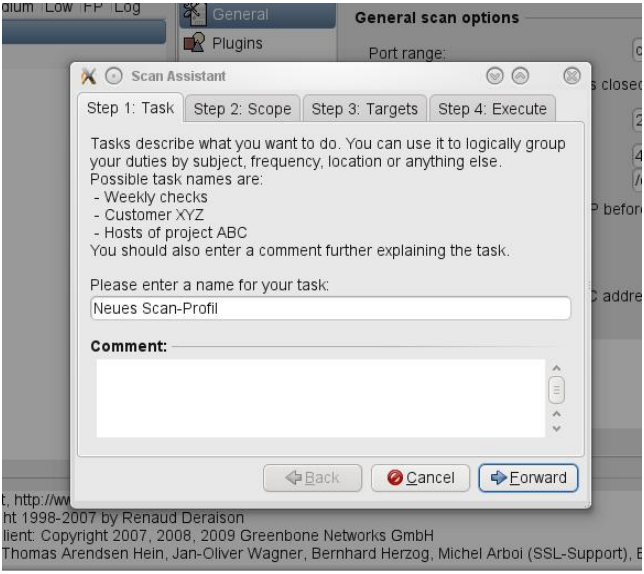
Ein Blick auf die der Kategorie *Web Servers* zugeordneten Plug-ins.

- Settings
- Slackware Local Security Checks
- SMTP problems
- SNMP
- Solaris Local Security Checks
- SuSE Local Security Checks

- Ubuntu Local Security Checks
- Useless services
- Web Servers
- Windows
- Windows: Microsoft Bulletins

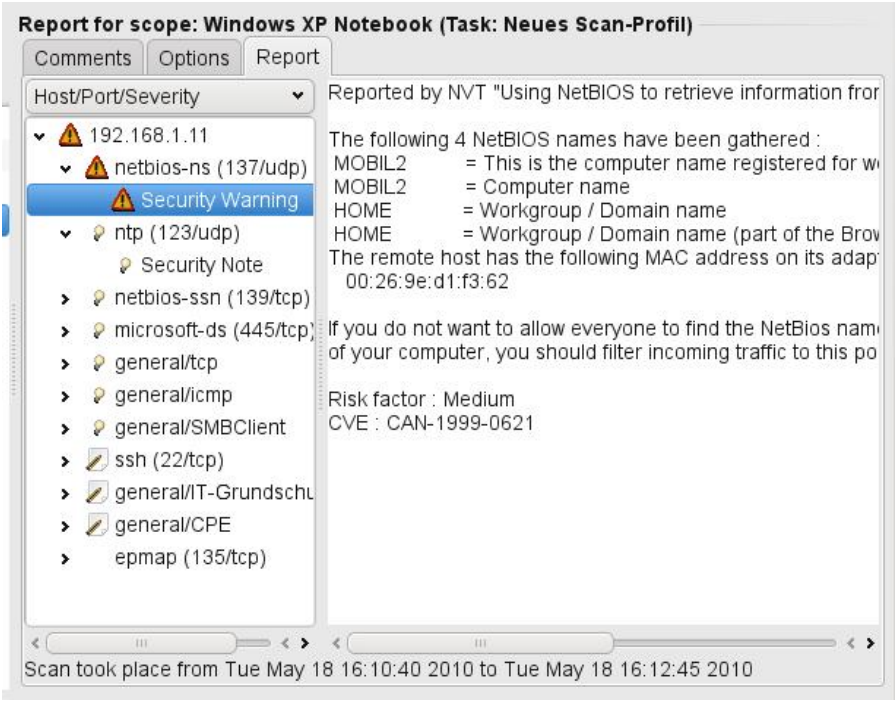
Wenn Sie eine Kategorie mit einem Klick öffnen, erkennen Sie, wie viele Plug-ins einer Kategorie zugeordnet sind. Mit einem Doppelklick auf einen Listeneintrag öffnen Sie die Detailinformationen zu den einzelnen Test-Skripts. Insbesondere über das Register *Options* können Sie eine Vielzahl von Einstellungen anpassen.

Den eigentlichen Scanvorgang leiten Sie mit einem Klick auf die Schaltfläche *Execute* ein. Den Scan-Vorgang zeigt der Client durch rotierende Punkte an. Die Ergebnisse präsentiert Ihnen der Client auf dem Register *Report*.



Der Scan-Assistent in Aktion.

Für das Erstellen von neuen Scan-Konfigurationen können Sie zwei Wege einschlagen: Sie können diese in der Scan-Liste anlegen oder aber zu dem Scan-Assistenten greifen, der über die Symbolleiste bzw. das Menü *File> Scan Assistant* verfügbar ist.



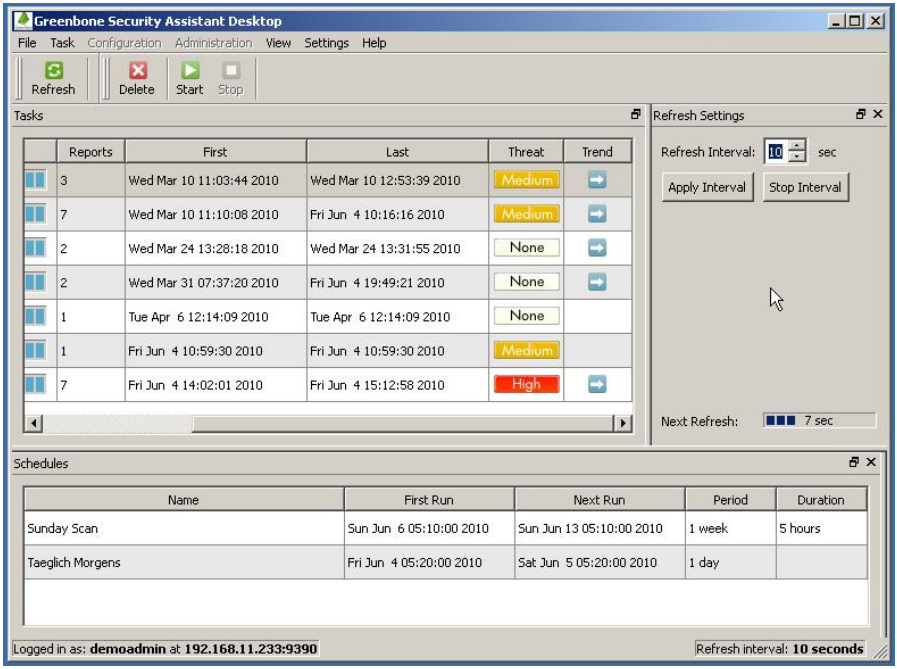
Eine typische Berichtausgabe.

1.3 Die Berichtausgabe

Die Berichtfunktion des OpenVAS-Clients finden Sie auf dem *Report*-Register des Clients. Der Bericht zeigt im linken Bereich die gescannten Systeme an, rechts finden Sie die zugehörigen Detailinformationen.

Kritische Ereignisse werden farbig gekennzeichnet und beispielsweise rot markiert. Die Berichtinfo liefert Ihnen in der Regel die notwendigen Informationen, um die (potenzielle) Schwachstelle schließen zu können.

Sie können die Berichte in zwei OpenVAS-spezifischen Formaten und als HTML-Daten speichern. Auch der Export nach ASCII und XML ist möglich. Auf die Details kommen wir in Kapitel 5 zu sprechen.



Auch für OpenVAS gibt es einen Windows-Client, der der Linux-Variante weitgehend entspricht.

Die Berichtausgabe unterliegt einer weiteren Einschränkung. Bislang ist ein direktes Schreiben der Berichtinformation in eine Datenbank nicht möglich.

Tipp: Professioneller Berichtexport

Der OpenVAS-Client unterliegt bei der Berichterstellung – wie schon seine Vorläufer – gewissen Einschränkungen. So kann man beispielsweise die Scan-Ergebnisse nicht editieren oder bearbeiten, bevor man den Bericht exportiert. Beim Windows-Client NessusWX, dem Vorläufer von NessusClient war dies möglich. Der Windows-Client konnte sogar Berichte in MySQL-Statements exportieren und unterstützte auch den Vergleich von Berichten.

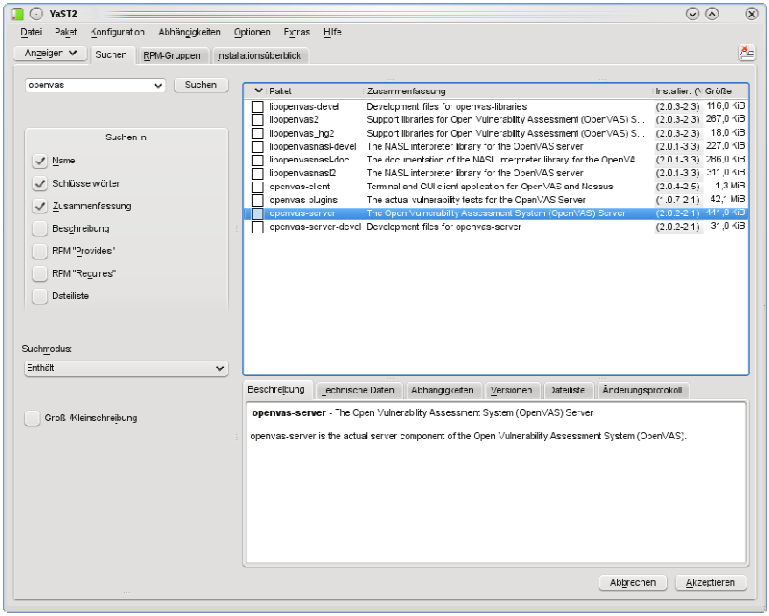
Leider ist das Tool von der Bildfläche verschwunden.

Damit haben Sie einen ersten Überblick, was Sie mit OpenVAS anstellen können. Schauen wir uns als Nächstes an, wie Sie den Server und den Client in Betrieb nehmen.

2 OpenVAS in Betrieb nehmen

Mit einem soliden Grundverständnis von OpenVAS sind Sie bestens für die Installation und Konfiguration gerüstet. Der Sicherheits-Scanner ist für die wichtigsten Linux-Distributionen verfügbar – sowohl als Server als auch als Client. Lediglich den Client gibt es auch in einer Windows-Variante. Sie können den OpenVAS-Scanner auf folgenden aktuellen Plattformen ausführen:

- Red Hat Enterprise Server
- Fedora Core
- SuSE Linux 9
- Debian
- FreeBSD



Auch mit YaST2 lässt sich OpenVAS installieren, allerdings nicht in der aktuellen Version.

Für die Installation, Inbetriebnahme und Nutzung genügen in der Regel grundlegende Linux-Kenntnisse und Grundkenntnisse im Bereich Sicherheit und Security Scanning. Sie müssen also keineswegs ein Linux-Crack sein, um erfolgreich mit OpenVAS arbeiten zu können. Allerdings liegt der Teufel wie so oft im Detail.

2.1 Voraussetzungen

Die Voraussetzungen für die Nutzung des Scanners sind recht bescheiden. Laut Angaben der Entwickler genügt ein Pentium, der mit 256 MB RAM ausgestattet ist, um ein lokales Netzwerk zu scannen. Bei großen Netzen sollten es allerdings schon 1 bis 4 GB sein.

Zunächst sei vorausgeschickt, dass Sie OpenVAS auch mithilfe einiger Paketmanager installieren können. Das ist beispielsweise unter verschiedenen RPM-basierten Linux-Distributionen wie OpenSuSE und Fedora der Fall. Wenn Sie mit Debian oder einer Ubuntu-Variante arbeiten, können Sie OpenVAS prinzipiell mit *apt-get* installieren.

Allerdings müssen Sie damit rechnen, dass diese nicht in aktuellen Versionen verfügbar sind. Daher empfiehlt es sich, OpenVAS aus den Binärpaketen zu installieren. Das ist sicherlich lästig und leider auch nicht zeitgemäß, aber der zuverlässigste Weg, um eine funktionierende OpenVAS-Installation aufzusetzen.

Bevor Sie sich an die Installation von OpenVAS machen, sollten Sie die Komponenten der Umgebung kennen. OpenVAS besteht aus vier Kernmodulen und drei optionalen Modulen.

Die Kernmodule im Überblick:

- **OpenVAS-Server**: Das Kernmodul von OpenVAS. Mit diesem Modul können Sie eine große Anzahl von Rechnern testen, und zwar in kurzer Zeit. Die Scans werden immer von dem Rechner ausgehen, auf dem der OpenVAS-Server ausgeführt wird. Es versteht sich von selbst, dass der Rechner, auf dem der OpenVAS-Server ausgeführt wird, die zu scannenden Zielrechner erreichen kann. Der Server ist – im Unterschied zu Nessus – nur für Linux-Betriebssysteme verfügbar.
- **OpenVAS-Libraries**: In diesem Modul sind die von OpenVAS genutzten Bibliotheken enthalten. Seit der Einführung von OpenVAS 3.0 ist OpenVAS-LibNASL in den Libraries aufgegangen. Diese Komponente enthält die eigentlichen Sicherheitstests (Network Vulnerability Tests, kurz NVTs). Die Tests, genauer die zugehörigen Skripts, sind in der Nessus Attack Scripting Language, kurz NASL, geschrieben. OpenVAS

hat sie also quasi von seinem Vorläufer geerbt. Inzwischen gibt es auch eigene Erweiterungen und Neuerungen.

- **OpenVAS-Plug-ins**: Dieses Modul stellt Ihnen eine Grundausstattung an Testskripts zur Verfügung. Sie sollten allerdings beachten, dass der Updatezyklus dieses Moduls nicht dazu gedacht ist, die Verfügbarkeit aktueller NVTs sicherzustellen. Wenn Sie aktuelle NVTs benötigen, sollten Sie einen NVT Feed abonnieren. Dazu später mehr.
- **OpenVAS-Client**: Der OpenVAS-Client ist die Komponente, in der Sie sich überwiegend bewegen. Sie steuert den OpenVAS-Server, verarbeitet die Scanergebnisse und präsentiert Ihnen diese die Scan-Ergerbnisse. Wichtig dabei: Der OpenVAS-Client kann auf (nahezu) jedem beliebigen Rechner ausgeführt werden. Der Client ist für Linux und Windows verfügbar. Er baut eine Verbindung zum OpenVAS-Server auf und steuert den Server. Sie können Client und Server natürlich auch auf dem gleichen System ausführen.

Neben diesen Kernmodulen gibt es vier weitere optionale Komponenten:

- **OpenVAS-Manager**: Dient dem Management der OpenVAS-Installation und verwendet für die Kommunikation das OMP (OpenVAS Management Protocol).
- **OpenVAS-Administrator**: Diese optionale Komponente dient der Administration der OpenVAS-Umgebung. Sie greift auf das OAP (OpenVAS Administration Protocol) zurück.
- **OpenVAS CLI**: Die Konsolenvariante (CLI, Command Line Interface) für Anwender, die lieber auf Konsolenebene arbeiten.
- **GSA**: Mit dem Greenbone Security Assistant steht Ihnen eine tolle webbasierte Umgebung für den Zugriff auf die OpenVAS-Umgebung zur Verfügung.

2.2 Installation des Basissystems

Ein Manko von OpenVAS ist sicherlich, dass es bislang im Vergleich zu seinem Vorläufer kein brauchbares Installationspaket gibt, mit dem Sie OpenVAS quasi in Minuten installieren können.

Wenn Sie OpenVAS zunächst nur evaluieren wollen, finden Sie auf der beiliegenden CD eine Live-Version, mit der Sie OpenVAS unmittelbar einsetzen können.

Achtung

Für die Installation von OpenVAS sind nicht unbedingt Root-Rechte erforderlich. Sie können das System auch als normaler Benutzer installieren.

Für die Grundkonfiguration müssen Sie zunächst die OpenVAS-Bibliotheken und den Scanner installieren, die sich hinter den Dateien *openvas-libraries-3.0.5.tar.gz* bzw. *openvas-scanner-3.0.2.tar.gz* verbergen. Außerdem müssen Sie den OpenVAS-Client installieren, damit Sie den OpenVAS-Server steuern können. Der Client verbirgt sich hinter dem Archiv *openvas-client-3.0.0.tar.gz.*

Beginnen Sie mit der Installation der OpenVAS-Bibliotheken. Zuvor sollten Sie sicherstellen, dass auf Ihrem Linux-System folgende Komponenten installiert sind:

- libglib 2.12 (oder höher)
- libgnutls 2.0 (oder höher)
- libpcap
- libgpgme 1.1.2 (oder höher)

Außerdem benötigen Sie folgende Tools:

- einen C-Compiler, beispielsweise gcc
- bison
- flex

Ganz wichtig: Stellen Sie auch sicher, dass die pcap-Komponenten auf Ihrem Server-System installiert sind.

Die Installation starten Sie mit folgendem Kommando:

```
./configure && make install
```

Beobachten Sie insbesondere die configure-Ausgaben, da hier womöglich fehlende Abhängigkeiten aufgeführt werden. Im Idealfall erfolgt die Installation erfolgreich und Sie können sich im nächsten Schritt der Installation des Scanners widmen.

Konnten die Bibliotheken erfolgreich auf Ihrem System installiert werden, wird nachstehende Erfolgsmeldung ausgegeben:

```
------------------------------------------------------------

 openvas-libraries has been successfully installed.

 Make sure that /usr/local/bin is in your PATH before you

 continue

 Be sure to add /usr/local/lib in /etc/ld.so.conf and type
'ldconfig'

------------------------------------------------------------
```

Um den Scanner zu installieren, wechseln Sie in das Verzeichnis, in das Sie das OpenVAS-Scanner-Archiv entpackt haben, und führen Sie dort der Reihe nach folgende Befehle aus:

```
./configure

make

make install
```

Der Scanner ist gleichbedeutend mit dem OpenVAS-Server.

Beobachten Sie auch hier wieder die configure-Ausgaben, um womöglich fehlende Abhängigkeiten zu erkennen.

Kann der Scanner korrekt installiert werden, wird zum Abschluss folgende Erfolgsmeldung ausgegeben:

```
------------------------------------------------------------

 openvas-scanner has been sucessfully installed.

 Make sure that /usr/local/bin and /usr/local/sbin are in
 your PATH before you continue.

 openvassd has been installed into /usr/local/sbin

------------------------------------------------------------
```

Damit ist das Herzstück des Systems eingerichtet.

2.2.1 Konfiguration

Die nächsten Schritte dienen der Konfiguration des OpenVAS-Servers. Die Kommunikation zwischen diesem und dem OpenVAS-Client wird per SSL abgesichert. Der Server muss aus diesem Grund dem Client ein Zertifikat anbieten können, mit dem er sich gegenüber dem Client authentifiziert.

Für das Erstellen eines Zertifikates stellt Ihnen OpenVAS ein entsprechendes interaktives Tool zur Verfügung. Sie finden das Werkzeug im Unterordner tools des OpenVAS-Scanner-Ordners. Wechseln Sie in den Ordner tools und führen Sie dort folgenden Befehl aus:

```
openvas-mkcert
```

Dieses Skript erzeugt zwei Zertifikate:

- Ein Zertifikat für eine lokale Zertifizierungsstelle (Certificate Authority, CA)
- Ein zweites Zertifikat für den OpenVAS-Server, das von dieser CA signiert ist und beim Verbindungsaufbau übermittelt wird.

Sollten Sie bereits über eine X.509-basierte Public-Key-Infrastruktur (PKI) verfügen, können Sie natürlich auch dieses Zertifikat verwenden. Kopieren Sie es einfach in den Standardordner */usr/local/var/lib/openvas/CA/*.

Das Skript verlangt von Ihnen verschiedene Angaben zur Gültigkeitsdauer, Ihrem Standard etc. Den Abschluss bildet die Ausgabe, dass die beiden Zertifikate erfolgreich erstellt wurden und wo diese abgelegt sind.

Hier ein Beispiel für den typischen Ablauf bei der Zertifikaterstellung:

```
# ./openvas-mkcert
-------------------------------------------------------------

       Creation of the OpenVAS SSL Certificate

-------------------------------------------------------------

This script will now ask you the relevant information to
create the SSL certificate of OpenVAS.

Note that this information will *NOT* be sent to anybody
(everything stays local), but anyone with the ability to con-
nect to your OpenVAS daemon will be able to retrieve this
information.
```

```
CA certificate life time in days [1460]: 1000
Server certificate life time in days [365]: 265
Your country (two letter code) [DE]: DE
Your state or province name [none]: Saarland
Your location (e.g. town) [Berlin]: Saarbruecken
Your organization [OpenVAS Users United]: brain-media

---------------------------------------------------------------
        Creation of the OpenVAS SSL Certificate
---------------------------------------------------------------

Congratulations. Your server certificate was properly
created.
/usr/local/etc/openvas/openvassd.conf updated
The following files were created:

. Certification authority:
   Certificate = /usr/local/var/lib/openvas/CA/cacert.pem
   Private key =
/usr/local/var/lib/openvas/private/CA/cakey.pem

. OpenVAS Server :
    Certificate =
/usr/local/var/lib/openvas/CA/servercert.pem
    Private key =
/usr/local/var/lib/openvas/private/CA/serverkey.pem

Press [ENTER] to exit
```

Mit einem abschließenden Klick auf ENTER schließen Sie diesen Teil der OpenVAS-Konfiguration ab.

Der OpenVAS-Server wird über einen Client angesprochen und gesteuert. Das setzt voraus, dass der Server zumindest einen einzigen Benutzer kennt, der ihn steuern darf.

Um einen ersten OpenVAS-Benutzer zu erzeugen, führen Sie ebenfalls in dem Ordner tools folgenden Befehl aus:

```
openvas-adduser
```

Auch bei diesem Kommando öffnet sich ein interaktiver Dialog, der von Ihnen verschiedene Angaben erfragt. Konkret verlangt der *adduser*-Befehl einige Eingaben wie den Benutzernamen und das Passwort. Hier ein Beispiel für den typischen Vorgang beim Erzeugen eines ersten Benutzers:

```
# openvas-adduser
Add a new openvassd user
--------------------
Login : admin
Authentication (pass/cert) [pass]:
Login password:
Login password (again):
User rules
----------
openvassd has a rules system which allows you to restrict the
hosts that admin has the right to test. For instance, you may
want him to be able to scan his own host only.


Please see the openvas-adduser(8) man page for the rules syn-
tax.


Enter the rules for this user, and hit ctrl-D once you are
done:
(the user can have an empty rules set)
Login :admin
Password :********
DN :
```

```
Rules :
Is that ok ? (y/n) [y]
User added.
#
```

Wenn Sie weitere Benutzer anlegen wollen, führen Sie wieder den Befehl *openvas-adduser* aus.

Wie Sie als Nächstes fortfahren, ist sicherlich von Ihren Kenntnissen und Anforderungen abhängig. Der eine Benutzer interessiert sich für weitere Konfigurationsmöglichkeiten, die der OpenVAS-Daemon zu bieten hat, ein anderer will lieber direkt erste Tests starten. Wieder ein anderer befasst sich vorzugsweise mit den verschiedenen Plug-in-Optionen.

2.2.2 Die OpenVAS-Konfigurationsdatei

Wenn Sie zu den eher technisch versierten Anwendern gehören, interessieren Sie sich vermutlich für die Einstellungen, die Ihnen die OpenVAS-Server-Konfigurationsdatei *openvass.conf* bietet. Damit Sie einen ersten Eindruck von den Möglichkeiten und vielfältigen Einstellungen bekommen, hier eine typische Beispieldatei mit den notwendigen Erläuterungen:

```
# Konfigurationsdatei des OpenVAS-Servers

# Pfad zu dem Sicherheitscheck-Ordner:
plugins_folder = /lib/openvas/plugins

# Maximale Anzahl an gleichzeitigen Tests.
max_hosts = 40

# Maximale Anzahl an Tests pro Host.
max_checks = 5

# Niedlichkeit. Bestimmt die Prozesspriorität. Wenn Sie diese
# Einstellung auf yes setzen, wird die Prozesspriorität
```

```
# auf 10 herabgesetzt.
be_nice = no

# Protokolldateien:
logfile = /var/log/openvas/openvas.messages

# Sollen alle Details einer Attacke
# protokolliert werden? Die Aktivierung
# ist sehr speicherplatzintensiv.
log_whole_attack = no

# Protokollierung der Plug-ins, die vom
# Server geladen werden?
log_plugins_name_at_load = no

# Dump-Datei für den Debug-Output.
dumpfile = /var/log/openvas/openvasd.dump

# Pfad zur Regeldatei.
rules = /etc/openvas/openvas.rules

# Benutzerdatenbank
users = /etc/openvas/openvasd.users

# CGI-Pfad für Prüfungen.
cgi_path = /cgi-bin:/scripts

# Port-Bereich, den es zu scannen gilt.
# Dabei bedeutet default, dass OpenVAS die
# Ports scannt, die in der Service-Datei
```

```
# gefunden werden.
port_range = default

# Optimiert die Tests (empfohlen).
optimize_test = yes

# Sprache der Plug-ins.
language = english

# Optimierung:
# Liest den Timeout-Wert für die Sockets.
checks_read_timeout = 5

# Ports, gegen die nicht zwei Plug-ins
# gleichzeitig attackieren sollen.
non_simult_ports = 139, 445

# Maximale Lebensdauer eines Plug-ins
# in Sekunden.
plugins_timeout = 320

# Sichere Checks, das bedeutet, dass
# Systeme nicht zum Absturz gebracht werden
# soll.
safe_checks = yes

# Aktiviert automatisch die abhängigen
# Plug-ins. Es gibt Tests, die auf
# einander aufbauen.
auto_enable_dependencies = yes
```

```
# Eine weitere Abhängigkeitseinstellung.
silent_dependencies = yes

# Bestimmt Hosts durch Ihre MAC- und nicht
# durch Ihre IP-Adresse.
use_mac_addr = no

# -- Einstellungen für die Knowledgebase --
# Sichert die Knowledgebase auf Festplatte.
save_knowledge_base = no

# Stellt die KB für jeden Test wieder her.
kb_restore = no

# Nur die Hosts, deren KB nicht vorhanden ist.
only_test_hosts_whose_kb_we_dont_have = no

# Nur die Hosts, deren KB bereits verfügbar ist.
only_test_hosts_whose_kb_we_have = no

# KB-Testwiederholung
kb_dont_replay_scanners = no
kb_dont_replay_info_gathering = no
kb_dont_replay_attacks = no
kb_dont_replay_denials = no
kb_max_age = 864000

# -- Ende der KB-Konfiguration
```

```
# Wenn Sie diese Option aktivieren, führt
# OpenVAS keine inkrementellen Scans durch
# (also 10.0.0.1, dann 10.0.0.2, 10.0.0.3
# und so weiter), sondern versucht die Last
# auf das gesamte Netzwerk zu verteilen
# (also 10.0.0.1, dann 10.0.0.127, dann
# 10.0.0.2, dann 10.0.0.128 und so weiter).
slice_network_addresses = no

# Sollen alle NASL-Skripts als signiert
# behandelt werden?
nasl_no_signature_check = no
```

Wie Sie voranstehender Beispielkonfiguration entnehmen können, sind über die OpenVAS-Konfigurationsdatei eine Vielzahl an Einstellungs- und Anpassungsmöglichkeiten geboten. Einige besonders wichtige bedürften der Erläuterung.

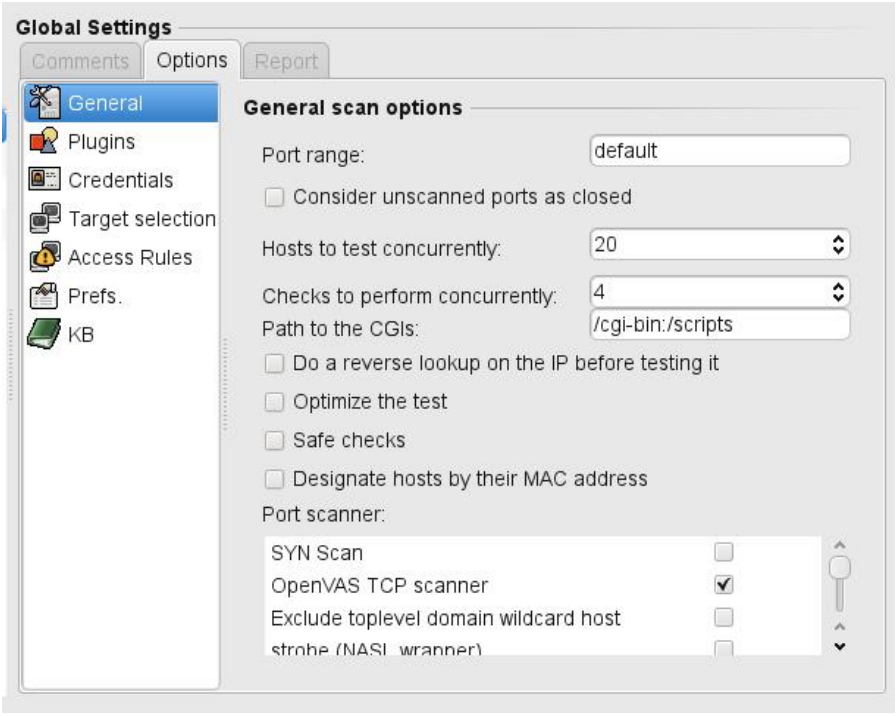
Die beiden wichtigsten Einstellungen für die Ausführung von Sicherheitsscans sind *max_hosts* und *max_checks*. Wie Sie obiger Beispielkonfiguration entnehmen können, werden in der Regel nicht mehr als 40 Systeme und 5 Tests pro Ziel gleichzeitig ausgeführt.

Für Anwender, die bereits mit Nessus gearbeitet haben, dürften die Unterschiede in der Nessus- und OpenVAS-Konfiguration interessant sein. Nessus weist zusätzlich folgende Konfigurationen auf:

```
# Plug-ins werden automatisch aktualisiert.
auto_update = yes

# Anzahl an Stunden zwischen zwei
# Update-Vorgängen.
auto_update_delay = 24

# Soll die Plug-in-Datenbank bei jedem
# Update bereinigt werden?
```

```
purge_plugin_db = no

# Drossle Scan, wenn die CPU überlastet ist?
throttle_scan = yes

# Können Benutzer Ihre Plug-ins uploaden?
plugin_upload = yes

# Suffixe der Plug-ins, die Benutzer
# hochladen können.
plugin_upload_suffixes = .nasl, .nasl3, .inc, .inc3, .nbin,
.audit

# Name der Benutzer, die Plug-ins aus
# der Ferne hochladen können.
admin_user = holger

# IP-Adresse, auf die der Server für
# eingehende Verbindungen hört.
listen_address = 0.0.0.0

# Dieser Abschnitt wird durch nessus-mkcert
# hinzugefügt.
cert_file=/opt/nessus//com/nessus/CA/servercert.pem
key_file=/opt/nessus//var/nessus/CA/serverkey.pem
ca_file=/opt/nessus//com/nessus/CA/cacert.pem

# Wenn Sie den privaten Schlüssel durch
# ein Passwort schützen wollen.
pem_password=password
```

```
# Erzwingt die Verwendung eines Client-
# Zertifikats.
force_pubkey_auth = yes
```



Auf dem Register *Options* des OpenVAS-Clients bestimmen Sie unter *General* z. B. die maximale Anzahl an Hosts, die der Server prüft.

Prinzipiell ist die Anzahl der Systeme, die ein OpenVAS-Server prüfen kann, von der Leistungsfähigkeit der eigenen Hardware abhängig. Daneben gibt es weitere Faktoren, die die Scan-Möglichkeiten beeinflussen, beispielsweise Sicherheitsrichtlinien für die Nutzung von internen Systemen etc. Hier ist etwa darauf zu achten, dass kritische Systeme nicht harten Attacken ausgesetzt werden.

Die Entwickler empfehlen generell, konservativ an die Sache heranzugehen. Wenn Sie OpenVAS auf einem typischen Linux-Server ausführen, so sollten Sie die Anzahl der gleichzeitigen Tests auf 20 setzen.

Ähnlich zurückhaltend sollten Sie bei der Anzahl an gleichzeitigen Tests pro Host sein. Beginnen Sie hier bei drei oder vier.

2.2.3 NVT Feeds konfigurieren

Mit dem sogenannten OpenVAS-NVT-Feed-Service steht Ihnen eine Sammlung am Test-Skripts zur Verfügung. Diese werden auch als NVTs bezeichnet (Network Vulnerability Tests). Sie besitzen die Dateierweiterung nasl oder inc. Für OpenVAS gibt es über 17.000 solcher Skripts, die Sie in Ihre OpenVAS-Installation herunterladen können.

Über den Feed-Service halten Sie diese Skripts auf dem neuesten Stand. Dabei werden nur neue und veränderte Skripts heruntergeladen, zusammen mit den jeweiligen Signaturen (asc-Dateien) und einer Datei mit Prüfsummen (md5sums).

Der Synchronisationsprozess basiert auf der RSYNC-Technologie. Die Signaturen sind nur dann relevant, wenn Sie Ihren OpenVAS-Server dazu konfiguriert haben, nur signierte Skripts auszuführen.

Um den Feed-Service nutzen zu können benötigen Sie neben dem Plug-in-Modul, inklusive dem Skript *openvas-nvt-sync*, die Programme *rsync* und *md5sum*.

Um nun Ihre lokale NVT-Sammlung mithilfe des Feed-Services auf den neuesten Stand zu bringen, arbeiten Sie folgende Schritte ab:

1. Stellen Sie zunächst sicher, dass das Synchronisationsskript installiert ist. Sie finden es standardmäßig in folgendem Verzeichnis:

   ```
   /usr/local/sbin/openvas-nvt-sync
   ```

   Prüfen Sie dabei insbesondere auch, ob die Variablen *NVT_DIR* und *FEED* die für Ihre Konfiguration korrekten Werte beinhalten.

2. Rufen Sie als Nächstes das Synchronisationsskript mit folgendem Kommando auf:

   ```
   # openvas-nvt-sync
   ```

Das Skript führt einen Datenabgleich mit dem angegebenen NVT-Feed-Service aus. Nach der Synchronisation werden die Prüfsummen aller synchronisierten Dateien überprüft.

3. Führen Sie dann einen Neustart des OpenVAS-Servers aus:

   ```
   # kill -1 PID
   ```

   Dabei steht PID für die Prozess-ID des openvassd-Prozesses. Fertig.

Vor dem Einsatz des Scanners müssen Sie das Skript zumindest einmal ausgeführt haben, da ansonsten nur ein Standardtest verfügbar ist. Sie können nach der Ausführung des Befehls *openvas-nvt-sync* den Abgleich bzw. dessen Fortschritt auf der Konsole verfolgen. Hier ein Ausschnitt aus einer typischen Ausgabe:

```
/usr/local/sbin/openvas-nvt-sync

[i] This script synchronizes an NVT collection with the
'OpenVAS NVT Feed'.

[i] The 'OpenVAS NVT Feed' is provided by 'The OpenVAS
Project'.

[i] Online information about this feed:
'http://www.openvas.org/openvas-nvt-feed.html'.

[i] NVT dir: /usr/local/lib/openvas/plugins

[i] Will use rsync

[i] Using rsync: /usr/bin/rsync

[i] Configured NVT rsync feed:
rsync://rsync.openvas.org:/nvt-feed

rsync server - Intevation GmbH, Germany

All transactions are logged. Mail problems to ad-
min@intevation.de.

Please look at /ftp/mirrors.txt for a list of download mir-
rors.


receiving file list ...
```

```
34975 files to consider
...
...
...
zeus_webserver_37829.nasl
        2660 100%    6.07kB/s    0:00:00 (xfer#34950, to-
check=21/34975)
zeus_webserver_37829.nasl.asc
         197 100%    0.45kB/s    0:00:00 (xfer#34951, to-
check=20/34975)
zml_cgi_traversal.nasl
        2032 100%    4.56kB/s    0:00:00 (xfer#34952, to-
check=19/34975)
zml_cgi_traversal.nasl.asc
         197 100%    0.44kB/s    0:00:00 (xfer#34953, to-
check=18/34975)
...
...
zone_alarm_fw_p67.nasl
        1744 100%    3.72kB/s    0:00:00 (xfer#34958, to-
check=13/34975)
zone_alarm_fw_p67.nasl.asc
         197 100%    0.42kB/s    0:00:00 (xfer#34959, to-
check=12/34975)
zone_alarm_local_dos.nasl
        1748 100%    3.70kB/s    0:00:00 (xfer#34960, to-
check=11/34975)
zone_alarm_local_dos.nasl.asc
         197 100%    0.42kB/s    0:00:00 (xfer#34961, to-
check=10/34975)
zope_37765.nasl
        3368 100%    6.95kB/s    0:00:00 (xfer#34962, to-
check=9/34975)
zope_37765.nasl.asc
```

```
         197 100%    0.41kB/s    0:00:00 (xfer#34963, to-
check=8/34975)

...

...

zyxel_http_pwd.nasl

        1696 100%    3.41kB/s    0:00:00 (xfer#34968, to-
check=3/34975)

zyxel_http_pwd.nasl.asc

         197 100%    0.40kB/s    0:00:00 (xfer#34969, to-
check=2/34975)

zyxel_pwd.nasl

        1565 100%    3.12kB/s    0:00:00 (xfer#34970, to-
check=1/34975)

zyxel_pwd.nasl.asc

         197 100%    0.39kB/s    0:00:00 (xfer#34971, to-
check=0/34975)


sent 559608 bytes  received 75219146 bytes  287585.40
bytes/sec

total size is 73245023  speedup is 0.97

[i] Checking dir: ok

[i] Checking MD5 checksum: ok
```

Nun stellt sich natürlich die Frage, wo Sie einen solchen Service finden. Das OpenVAS-Projekt stellt unter *rsync://rsync.openvas.org:/nvt-feed* einen NVT-Feed-Service zur Verfügung. Dieser ist auch im Synchronisationsskript voreingestellt.

Sie können auch für einen automatischen Abgleich mit einem NVT-Feed-Service sorgen. Dazu erstellen Sie ein Skript mit der Bezeichnung *openvas-update* und speichern es im Verzeichnis */usr/local/bin*:

```
#!/bin/sh

temp=‘tempfile‘

openvas-nvt-sync 2>&1> $temp
```

```
if [ $? -ne 0 ]
then
cat $temp
fi
rm $temp
if [ -f /var/lib/run/openvasd.pid ]
then
pid=‘cat /var/lib/run/openvasd.pid‘
kill -1 $pid 2>/dev/null
fi
```

Erweitern Sie Ihre Crontab-Konfiguration um folgende Zeile:

```
25 4 * * * root /usr/local/bin/openvas-update
```

Sie müssen sich in nun Zukunft nicht mehr darum kümmern, dass Ihre Skripts auf dem neuesten Stand sind. Wie Sie eigene Skripts erstellen können, erfahren Sie in Kapitel 9.

2.2.4 OpenVAS-Server starten und anhalten

Sind Sie mit der Grundkonfiguration des Systems zufrieden, können Sie den OpenVAS-Server in Betrieb nehmen. Dazu führen Sie den bereits vorgestellten Startbefehl aus:

```
# /usr/local/sbin/openvassd
```

Nach dem Start des Servers sollte auch die Meldung ausgegeben werden, dass die Plug-ins, also die Test-Skripts, geladen sind.

Das Laden der Skripts kann durchaus einige Zeit in Anspruch nehmen, wenn Sie die Plug-ins das erste Mal nutzen. Sie können den Ladevorgang auf der Konsole verfolgen. Eine typische Ausgabe sieht wie folgt aus:

```
Loading the plugins ... 4896 (out of 17442)
```

Sind alle Plug-ins geladen, wird die simple Meldung ausgegeben, dass alle Test-Skripts geladen sind:

```
All plugins loaded
```

Der Ladevorgang dauert beim ersten Laden besonders lange, bei der späteren Ausführung wird er deutlich schneller ausgeführt.

Sie sind natürlich herzlich eingeladen, sich an der Entwicklung der Plug-ins zu beteiligen.

Soll der Scanner bei jedem Systemstart automatisch gestartet werden, so fügen Sie obigen Befehl in das Startskript ein:

```
/etc/rc.d/rc.local
```

Auf die Ausgabe beim Server-Start müssen wir nicht mehr eingehen. Sie ist bereits zuvor beschrieben worden. Wichtig ist, dass Sie OpenVAS als Root ausführen, denn nur Root kann den Scanner starten und beenden.

Auch das Stoppen des Servers wurde bereits erwähnt. Hier der Vollständigkeit halber noch einmal der zugehörige Befehl:

```
# killall openvassd
```

2.2.5 Kommandozeilenoptionen des OpenVAS-Servers

Für die Steuerung des OpenVAS-Servers stehen Ihnen einige weitere Kommandozeilenoptionen zur Verfügung. Hier ein Beispiel:

```
# /usr/local/sbin/openvassd [-fhv] [-c <konfigdatei>] [-p
<port-nummmer>] [-a <adresse>] [-S <ip[,ip,...]>]
```

Nachstehende Tabelle fasst die Optionen und ihre Bedeutung zusammen:

Option	Beschreibung
-c <konfigdatei>	Beim Start können Sie auch eine alternative Konfigurationsdatei angeben, um mit einer anderen Konfiguration ergänzende Informationen zu erzielen.
-a <adresse>	Hier können Sie beim Start bereits die IP-Adresse bestimmen, auf die der Server anspricht.
-S <ip1, ip2, …>	Hiermit erzwingen Sie bestimmte IP-Adressen.
-p <port-nummer>	Wenn Sie eine alternative Port-Nummer verwenden wollen, so definieren Sie diese mit diesem Schalter.
-D	Hiermit starten Sie den Server im Daemon-, also Hintergrundmodus.
-v	Dieser Schalter gibt die Versionsnummer aus und beendet sich.
-h	Dieser Schalter gibt die Hilfetexte aus.

3 Der OpenVAS-Client

Nachdem Sie den aktuellen OpenVAS-Server installiert haben, benötigen Sie einen Client, um diesen zu steuern. Die OpenVAS-Umgebung setzt, wie bereits erwähnt, auf eine typische Client-Server-Architektur. Sie können den OpenVAS-Server von jedem beliebigen Client aus steuern – sofern ein entsprechender Client verfügbar ist. Sie können Ihren Linux-basierten Server also beispielsweise von einem Linux- oder Windows-Client aus steuern. Leider wird es für Mac bis auf Weiteres keinen Client geben.

Wenn Sie den Server von einem Linux-System aus steuern wollen – womöglich auch von dem gleichen System, auf dem der Server installiert ist –, so müssen Sie den offiziellen OpenVAS-Client installieren. Alternativ ist auch der Einsatz des GSA möglich, allerdings nur in Verbindung mit dem OpenVAS-Manager. Darauf kommen wir in Kapitel 6 zu sprechen.

3.1 OpenVAS-Client installieren

Die Installation des OpenVAS-Clients ist wirklich ein Kinderspiel. Laden Sie sich das Client-Archiv *openvas-client-3.0.0* herunter, entpacken Sie es und führen Sie der Reihe nach folgende Kommandos aus:

```
./configure
```

```
make
```

```
make install
```

Kann der Scanner korrekt installiert werden, wird zum Abschluss folgende Erfolgsmeldung ausgegeben:

```
-----------------------------------------------------------
 OpenVAS-Client has been successfully installed.
-----------------------------------------------------------
```

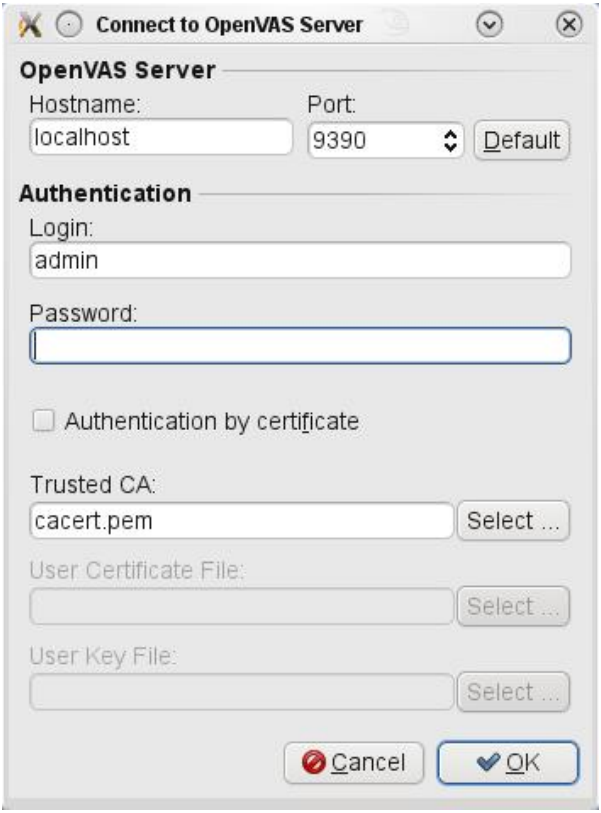
Herzlichen Glückwunsch: Sie haben OpenVAS vollständig installiert – zumindest soweit, dass Sie mit der Umgebung arbeiten können. Sie starten den Client mit folgendem Befehl:

```
OpenVAS-Client
```

3.2 Erstkonfiguration mit dem OpenVAS-Client

Nachdem Sie OpenVAS in Betrieb genommen haben, können Sie sich als Nächstes der Server-Konfiguration widmen. Dazu verwenden Sie, wie bereits erwähnt, den OpenVAS-Client.

Stellen Sie zunächst sicher, dass der von Ihnen gewünschte Server verfügbar ist, und starten Sie dann den Client. Dieser präsentiert Ihnen seine übersichtliche Schnittstelle mit zwei Bereichen: Links die Scans, rechts die Einstellung für die Scans bzw. die Berichte und Kommentare.



Über den OpenVAS-Client rufen Sie den Verbindungsdialog auf und stellen die Verbindung zum OpenVAS-Server her.

Als Nächstes bestimmen Sie über den Connection-Dialog, mit welchem OpenVAS-Server die Tests durchgeführt werden sollen. Klicken Sie dazu in der Symbolleiste des Clients auf das Connect-Icon.

Bei der Erstinstallation finden Sie im Verbindungsdialog den Eintrag *localhost* für den Verbindungsaufbau. Erstellen Sie gegebenenfalls einen oder weitere neue Verbindungseinträge.

Das Erstellen einer neuen Verbindung ist einfach. Im Eingabefeld *Host name* geben Sie die IP-Adresse bzw. den Hostnamen des OpenVAS-Servers an. Der OpenVAS-Server verwendet standardmäßig den Port 9390.

Dann geben Sie den Benutzernamen und das Passwort für den Verbindungsaufbau an. Beide haben Sie bei der Einrichtung des OpenVAS-Servers mit *openvas-adduser* eingerichtet. Wenn Sie weitere Benutzer benötigen, erzeugen Sie diese einfach auf dem OpenVAS-Serversystem entsprechend obiger Anleitung.

Für die Authentifizierung des Clients können Sie auch Zertifikate verwenden. Aktivieren Sie dazu die SSL-Option *Authentication by certificate* und geben Sie die CA sowie die Zertifikats- und Schlüsseldateien an.

Im Eingabefeld *Trusted CA* geben Sie das Zertifikat einer Certificate Authority (CA) an, der Sie vertrauen. Damit wird überprüft, ob Sie sich mit einem vertrauenswürdigen OpenVAS-Server verbinden. Dieser Check wird für die Paranoia Level 2 und 3 durchgeführt, nicht jedoch für den Level 1. Level 1 kann allerdings in der Konfigurationsdatei .openvasrc konfiguriert werden.

Wenn Sie eine Verbindung zu einem entfernten statt zu einem lokalen Server aufbauen, benötigen Sie eine Kopie des CA-Zertifikats. Legen Sie es in Ihrem Home-Verzeichnis ab.

Wenn Sie die Option *Authentication by certificate* aktivieren, benötigen Sie ein für Sie erstelltes Schlüssel-Zertifikat-Paar. Das wird üblicherweise vom Administrator des OpenVAS-Servers mit den entsprechenden Skripten erzeugt.

Sie erhalten zwei Dateien:

- Benutzer-Zertifikat
- Benutzer-Schlüssel

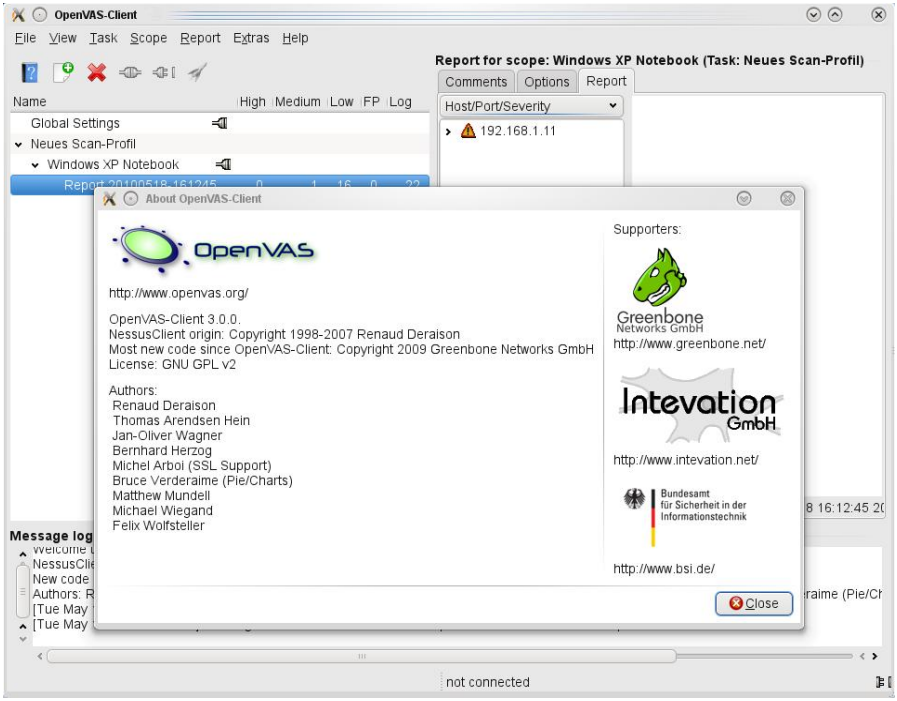
Der Schlüssel kann mit einem Passwort versehen sein. Ist das der Fall, geben Sie es beim Verbindungsaufbau an.

Mit einem Klick auf die Schaltfläche *OK* stellen Sie die Verbindung zwischen Client und Server her. Ein kleiner Connection-Dialog zeigt das Laden der Plug-ins an und öffnet dann den OpenVAS-Client. In der Plug-in-Auswahl erkennen Sie, dass die Testskript-Kategorien verfügbar sind.

Dass eine Verbindung zwischen dem OpenVAS-Client und dem Server besteht, erkennen Sie auch daran, dass das Verbindungs-Icon der Symbolleiste hellgrau hinterlegt ist, das Disconnect-Icon hingegen dunkelgrau.

Bevor wir auf die unzähligen Scan-Einstellungen zu sprechen kommen, sollten Sie noch die Funktionen des OpenVAS-Menüs kennen. Der Client stellt Ihnen sechs Menüs zur Verfügung:

- **File**: Hier finden Sie typische dateibezogene Funktionen wie das Öffnen, Speichern etc. Auch der Aufruf des Scan-Assistenten, der Programmeinstellungen und das Beenden des Programms ist hier möglich.
- **View**: Dieses Menü erlaubt das Ein- und Ausblenden der Symbolleiste und des Protokoll-Bereichs unterhalb der Profileinstellungen und Profilverwaltung.
- **Task**: Dient dem Erstellen, dem Umbenennen und dem Löschen von Scan-Aufgaben. Hinter diesen Aufgaben verbergen sich die Scan-Profile.
- **Scope**: Hier finden Sie die Funktionen für die Ausführung von Scans, das Erstellen neuer Profile sowie für das Öffnen und Speichern.
- **Report**: In diesem Menü finden Sie die berichtrelevanten Funktionen. Sie können Berichte beispielsweise umbenennen und löschen. Auch die Import- und Exportfunktionen sind hier zu finden.
- **Extras**: Dieses Menü ist für das Zusammenspiel mit Drittanwendungen zuständig. Hier können Sie beispielsweise den SLAD-Install-Manager aufrufen.
- **Help**: Das letzte Menü erlaubt den Zugriff auf die Dokumentation des OpenVAS-Teams und die typische Produktinformation.



Der Info-Dialog des OpenVAS-Clients verrät Ihnen die exakte Produktversion.

3.3 OpenVAS-Begriffswelt

Damit Sie effektiv mit dem OpenVAS-System arbeiten können, sollten Sie auch mit den Begrifflichkeiten des Systems vertraut sein. Dieses unterscheidet sich von seinem Vorgänger und vermutlich auch von anderen vergleichbaren Lösungen.

Um OpenVAS vernünftig einsetzen zu können, sollten Sie die beiden Begriffe Task (Aufgabe) und Scope (Bereich) und deren OpenVAS-spezifische Bedeutungen kennen.

Unter dem Oberbegriff Aufgabe werden alle Aktionen einer Aufgabenstellung zusammengefasst. Sie könnten beispielsweise für die Tests Ihrer lokalen Infrastruktur die Aufgabe „Test der eigenen Infrastruktur“ anlegen. Wenn Sie Sicherheitstests für Ihre Kunden durchführen, könnten Sie eine weitere Aufgabe als Kunde A oder als Kunde B bezeichnen. Sie können Aufgaben auch als Projektbezeichnungen verstehen. Wie Sie diese Funktion einsetzen, bleibt letztlich Ihnen überlassen.

Sie können einer Aufgabe auch Kommentare zuordnen, die Sinn und Zweck detailliert beschreiben. Sie können einer Aufgabe dabei beispielsweise auch Zielsetzungen, Scan-Häufigkeiten oder auch vertragliche Informationen zuweisen.

Fürs Verständnis wichtig: Aufgaben besitzen keine Scan-Optionen und auch keine Berichte. Sie stellen lediglich eine Sammlung von Bereichen dar. Sie können in einer Aufgabe eine Baumstruktur anlegen.

Der zweite wichtige Begriff ist Bereich. Darunter versteht man eine Teilaufgabe, konkret einen bestimmten Sicherheits-Scan. Wichtig dabei: Weisen Sie den Tests aussagekräftige Bezeichnungen zu. Diese könnten beispielsweise lauten „Test aller produktiven Webserver“ oder „Tests der Windows-Systeme der Abteilung Marketing“.

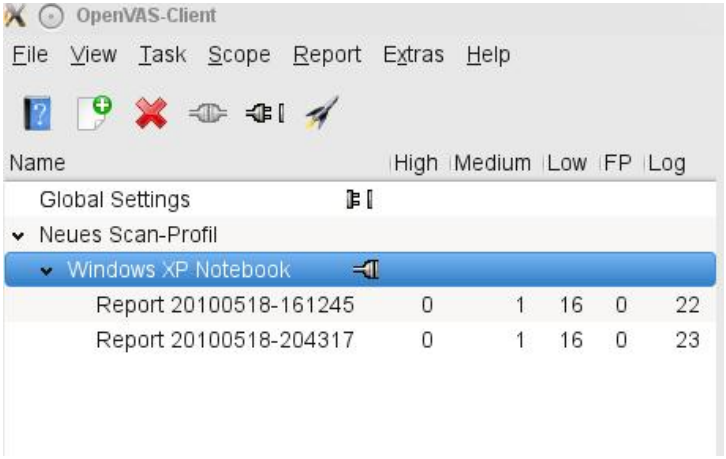
Auch Bereichen können Sie Kommentare zuordnen, um beispielsweise deren Zielsetzung innerhalb einer Aufgabe zu beschreiben. Ein Bereich ist außerdem durch einen vollständigen Satz an Scan-Einstellungen definiert. Auf die vielfältigen Konfigurationsmöglichkeiten kommen wir später noch zu sprechen.

Sie können für jeden Bereich eine Verbindung zu einem OpenVAS-Server aufbauen. Dabei wird die Auswahl der Plug-ins als Teil der Einstellungen im Client angezeigt. Neben jedem Bereichseintrag zeigt die Profilverwaltung den Verbindungsstatus durch das Connect- bzw. Disconnect-Icon an.

Haben Sie sich für die Zuordnung eines Servers zu einem Bereich entschieden, damit jeder Server unterschiedliche Plug-ins anbieten kann. Sie wissen ja bereits, dass nach dem Verbindungsaufbau aufseiten des Clients die Liste der verfügbaren Plug-ins angezeigt wird.

Bereiche sind außerdem durch Berichte gekennzeichnet. Jeder Bereich kann dabei eine Sammlung von Berichten umfassen. Immer dann, wenn Sie einen Scan-Vorgang erfolgreich ausgeführt haben, erzeugt OpenVAS einen Bericht und ordnet diesen dem Bereich zu.

Dabei ist zu beachten, dass Änderungen an einem Bereich immer erst dann gespeichert werden, wenn ein Scan gestartet wird.



Ein Bereich weist zwei Berichte auf.

Der OpenVAS-Client führt die Aufgaben und Bereiche, aber auch die erzeugten Berichte in einer übersichtlichen Tabelle auf. Das Schöne daran: In den Spalten *High*, *Medium*, *Low*, *FP* und *Log* erfahren Sie direkt mehr über die durchgeführten Tests und können unmittelbar erkennen, ob kritische Sicherheitslücken aufgedeckt wurden.

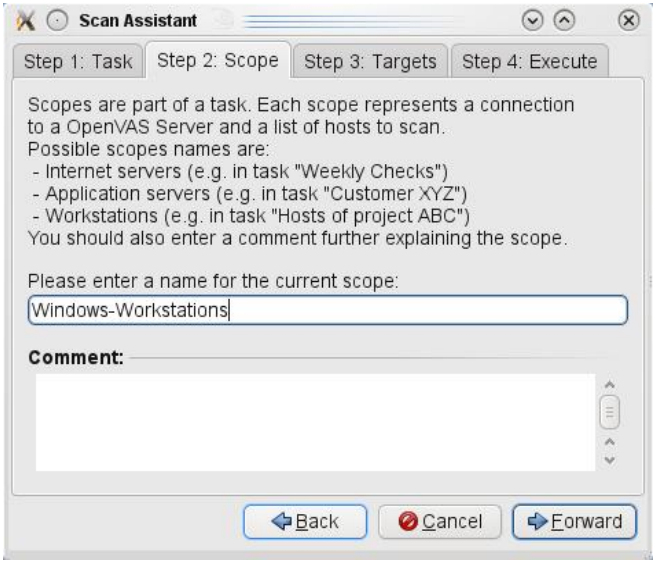
Da die Bereiche und Berichte den Aufgaben hierarchisch im linken Fensterbereich zugeordnet werden, ist es ein Leichtes, sich durch die Hierarchie zu hangeln, um die gewünschten Informationen anzuzeigen.

3.4 Der Scan-Assistent

Wenn Sie die ersten Gehversuche mit OpenVAS unternehmen, so werden Sie sich vermutlich schnell mit dem Scan-Assistenten anfreunden. Der hilft Ihnen, die wichtigen Einstellungen einer Aufgabe anzulegen. Wie es für einen Assistenten gehört, führt er Sie durch die vier notwendigen Schritte.

Sie rufen den Assistenten mit einem Klick auf das erste Icon der OpenVAS-Client-Symbolleiste (das blaue Buch mit weißem Fragezeichen) oder aber mit dem Befehl *File> Scan Assistant* auf.

Im ersten Dialog weisen Sie Ihrer neuen Aufgabe eine Bezeichnung zu.



Der zweite Schritt des Assistenten dient dem Anlegen des Bereichs.

Wie beim ersten Schritt können Sie auch bei Schritt 2 einen Kommentar in dem Comment-Feld hinterlegen.

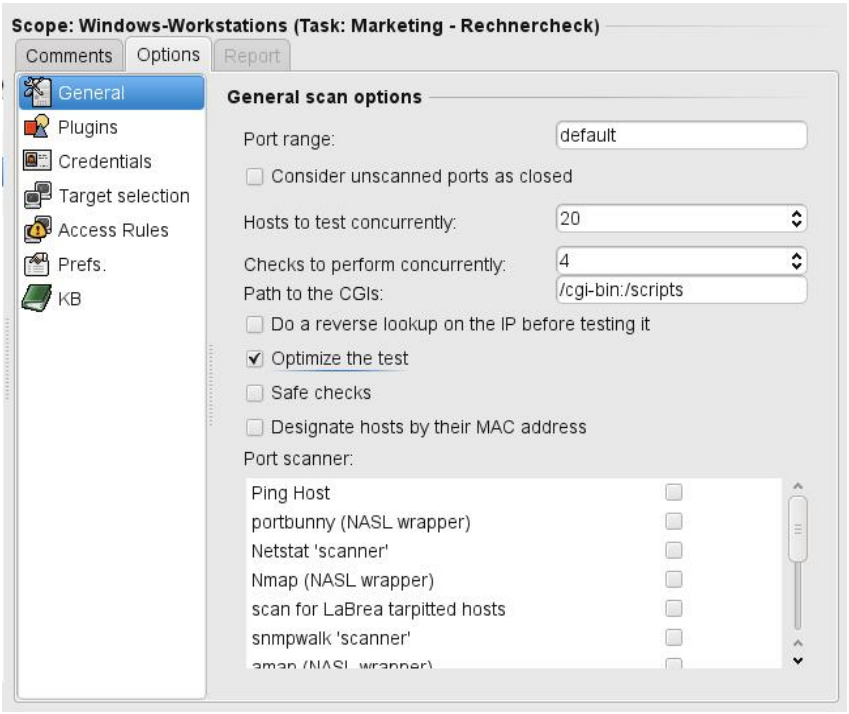
Mit einem Klick auf die *Forward*-Schaltfläche gelangen Sie zum dritten Schritt: der Definition des bzw. der Zielsysteme, die Sie Tests unterziehen wollen. Geben Sie dazu in dem Eingabefeld entweder den Hostnamen, die IP-Adresse bzw. den Adressbereich oder die IP-Adresse samt virtuellem Host an.

Wenn Sie mehrere Adressen, Hostnamen oder Adressbereiche scannen wollen, trennen Sie die Einträge durch Komma voneinander.

Mit einem weiteren Klick auf die Schaltfläche *Forward* gelangen Sie zum letzten Schritt. Der weist Sie darauf hin, dass der Assistent alle notwendigen Informationen für die Durchführung eines Scans besitzt und dass die Scan-Vorgänge in einem Fortschrittsfenster angezeigt werden. Mit einem Klick auf die *Execute*-Schaltfläche starten Sie den Scan-Vorgang.

3.5 Die Scan-Optionen

Steht die Verbindung zwischen dem OpenVAS-Client- und Server, können Sie für den Scan-Vorgang eine Fülle an Einstellungen bearbeiten – und zwar für jeden Bereich eigene Einstellungen.



Die Konfiguration einer Scan-Policy erfolgt auf sieben Registern.

Für die Konfiguration Ihrer Scan-Einstellungen stehen Ihnen sieben Konfigurationsbereiche zur Verfügung: *General*, *Plugins*, *Credentials*, *Target Selection*, *Access Rules*, *Prefs* und *KB*.

Zunächst sollten Sie sich den Einstellungen des Registers *General* widmen. Wenn Sie ein wenig mit OpenVAS vertraut sind, können Sie natürlich auch einen anderen Weg einschlagen, doch zunächst sollten Sie die Scan-Optionen kennenlernen.

Markieren Sie einen *Policy*-Eintrag und öffnen Sie dessen Einstellungen mit einem Klick auf die *Edit*-Schaltfläche unterhalb der Policy-Liste.

3.5.1 Die allgemeinen Scan-Optionen

Die erste Konfiguration des Registers *General* trägt die Bezeichnung *Port range*, zu Deutsch Port-Bereich. Hier bestimmen Sie die Auswahl der Ports, die durch den OpenVAS Server gescannt werden sollen. Sie können dabei entweder einen individuellen Port-Bereich angeben, beispielsweise in der Form 1030-1400 oder aber die Ports als kommaseparierte Liste in der Form 25,110,1024. Sie können auch beide Formen miteinander kombinieren.

Wenn Sie keine Portscans durchführen wollen, geben Sie den Wert -1 ein. Der Standardeintrag *default* bedeutet, dass die in der OpenVAS-Dienstedatei (*openvas-services*) angegebenen Ports gescannt werden.

OpenVAS greift bei seinen Portscan-Tests auch auf altbekannte Werkzeuge wie Netstat & Co. zurück und ist darauf angewiesen, dass diese korrekt arbeiten. Soweit es die Scans betrifft, hat die Option *Consider unscanned ports as closed* einen erheblichen Einfluss auf die Berichtausgabe. Wenn Sie diesen Schalter aktivieren, geht OpenVAS davon aus, dass nicht gescannte Ports sicher sind. Dass das nicht notwendigerweise der Fall sein muss, versteht sich von selbst.

Im Eingabefeld *Hosts to test concurrently* geben Sie die Anzahl an Hosts an, die gleichzeitig getestet werden sollen. Beachten Sie bei der Anpassung dieser Konfiguration, dass der OpenVAS-Server für einen Scan *max_hosts* x *max_tests* Prozesse startet. Ein deutliches Hochsetzen führt dabei schnell zu einer sehr hohen Systembelastung.

Mit dem Eingabefeld *Checks to perform concurrently* legen Sie fest, wie viele Tests zur gleichen Zeit durchgeführt werden sollen.

Mit dieser Einstellung bestimmen Sie also die Anzahl an Tests, die durch den Server gleichzeitig gegen jeden Host gestartet werden. Beachten Sie auch hier, dass der OpenVAS-Server für einen Scan *max_hosts* x *max_tests* Prozesse erzeugt.

Prinzipiell ist die Anzahl der Systeme, die ein OpenVAS-Server prüfen kann, von der Leistungsfähigkeit der eigenen Hardware abhängig. Daneben gibt es weitere Faktoren, die die Scan-Möglichkeiten beeinflussen, beispielsweise Sicherheitsrichtlinien für die Nutzung von internen Systemen etc. Hier ist etwa darauf zu achten, dass kritische Systeme nicht harten Attacken ausgesetzt werden.

Eine weitere Besonderheit von OpenVAS: Sie können mehrere Pfade, also beispielsweise */cgi-bin*, */cgis*, */home-cgis* etc. nach CGIs durchsuchen. Wenn Sie mehrer Pfade angeben wollen, so trennen Sie diese durch Doppelpunkte.

Mit der Option *Do a reverse lookup on the IP before testing it* können Sie vor dem Test eine rückwärtige DNS-Namensauflösung der IP-Adresse durchführen. Wenn Sie diese Option einschalten, führt der Server die DNS-Rückwärtssuche (DNS reverse lookup) durch, bevor der Test gestartet wird. Das Aktivieren verlangsamt die Testausführung natürlich, liefert Ihnen aber auch eine womöglich interessante Zusatzinformation.

Verschiedene Test-Skripts sind so intelligent angelegt, dass Sie nur unter bestimmten Voraussetzungen ausgeführt werden. Mögliche Voraussetzungen sind beispielsweise, dass sich die Daten in der Wissensdatenbank befinden oder dass ein angegebener Port offen ist. Mit der Option *Optimize the test* sorgen Sie für dieses intelligente Verhalten.

Für das Testen von Verwundbarkeiten eines Systems gibt es zwei Ansätze: eindringendes Scannen und nicht-eindringendes Scannen. Im ersten Fall sendet man an den jeweiligen Dienst Daten, die die Schwachstelle ausnutzen und beispielsweise ein System zum Absturz bringen können. Bei der zweiten Methode sendet man Anfragen an den Dienst, die die Schwachstelle verifizieren, den Dienst aber selbst nicht lahm legen oder auf eine andere Art beschädigen.

OpenVAS unterstützt beide Verfahren. Um die ungefährlichere Variante zu wählen, aktivieren Sie zunächst die Option *Safe check*. Diese Option sorgt auch dafür, dass nicht unnötige Informationen in den Berichten landen. Für den Administrator ist es besonders schwierig, sie als solche zu identifizieren. Es dauert nicht nur lange, bis sie erkannt und behoben sind, sondern sie können sogar ganze Testreihen (auch solche über einen längeren Zeitraum hinweg) infrage stellen. Also muss man Wege finden, wie man mit solchen Problemen umgeht. Auch hierfür ist OpenVAS gerüstet.

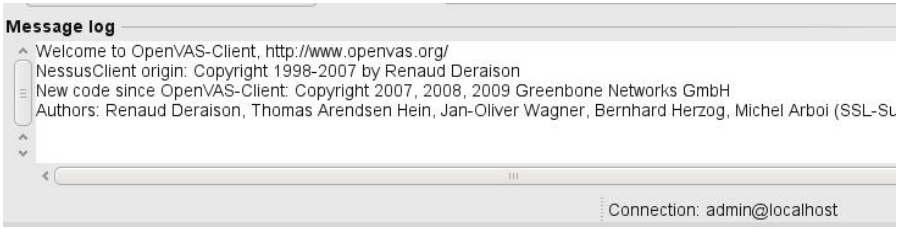
Die Option *Designate hosts by their DNS name* hat Auswirkungen auf die Scan-Performance – gerade bei sehr umfangreichen Testvorgängen. Normalerweise gibt OpenVAS die Ergebnisse seiner Tests nach IP-Adressen sortiert aus. Wenn Sie die Sortierung nach Hostnamen vorziehen, aktivieren Sie diese Option. Sie hat auch positive Auswirkungen auf die Scan-Vorgänge, wenn Sie viele Systeme den Sicherheitstests unterziehen. Der Grund für die schnellere Ausführung: Die DNS-

Einträge werden nicht sauber aktualisiert. Das kann pro Host-Adresse bis zu 10 Sekunden Zeit einsparen.

Bei Nessus konnte man bei allgemeinen Einstellungen auch die Protokollierung ein- bzw. ausschalten. In OpenVAS ist diese Konfigurationsmöglichkeit über den OpenVAS-Client leider nicht mehr gegeben. Aber seien Sie beruhigt: Auch OpenVAS legt Protokolldateien an, um genau zu sein zwei: *openvasd.messages* und *openvassd.dump*. Die eine enthält die Protokolle der durchgeführten Scan, die andere die Protokolle der Server-Aktivitäten.

In die Datei *openvasd.messages*, die standardmäßig unter */var/local/var/log/openvas/openvasd.messages* abgelegt ist, werden die OpenVAS-Servermeldungen geschrieben. Indem Sie den Wert in der Konfigurationsdatei auf *syslogd* setzen, verwendet der OpenVAS-Server den Syslogd-Dienst.

Die wichtigsten Protokollinformationen zeigt Ihnen ja auch der Bereich *Message Log* des OpenVAS-Clients an.



Der *Message Log*-Bereich des OpenVAS-Clients zeigt die wichtigsten Ereignisse an.

Wie Sie voranstehender Abbildung entnehmen können, zeigt die OpenVAS-Client-Statusleiste außerdem an, welcher Benutzer mit welchem OpenVAS-Server verbunden ist.

Wenn Sie die Details des Scan-Vorgangs aufzeichnen wollen, setzen Sie die Option *log_whole_attack* der OpenVAS-Konfigurationsdatei von der Standardeinstellung *no* auf *yes*. In der Standardeinstellung *no* werden lediglich der Start- und Endzeitpunkt des Scans protokolliert. Wenn Sie die Einstellung auf *yes* setzen, zeichnet der Server den Ablauf detaillierter und protokolliert beispielsweise auch die Zeiten, die die einzelnen Tests zur Ausführung benötigt haben.

Hier ein Ausschnitt aus einer typischen Server-Protokolldatei:

```
[Mon May 17 20:44:51 2010][15718] openvassd 3.0.2. started
```

```
[Mon May 17 21:02:17 2010][15718] received the TERM signal

[Mon May 17 21:02:23 2010][15777] openvassd 3.0.2. started

[Mon May 17 21:35:25 2010][27910] bad login attempt from ::1

[Mon May 17 21:35:34 2010][32469] bad login attempt from ::1

[Mon May 17 21:41:07 2010][4722] bad login attempt from ::1

[Mon May 17 21:42:47 2010][7821] scheduler: synscan.nes de-
pends on ping_host.nasl which could not be found, thus this
dependency is not considered for execution sequence

[Mon May 17 21:42:47 2010][7821] scheduler: open-
vas_tcp_scanner.nes depends on ping_host.nasl which could not
be found, thus this dependency is not considered for execu-
tion sequence

[Mon May 17 21:42:47 2010][7821] user admin starts a new
scan. Target(s) : localhost, with max_hosts = 20 and
max_checks = 4

[Mon May 17 21:42:47 2010][7821] user admin : testing
127.0.0.1 (::ffff:127.0.0.1) [7894]

[Mon May 17 21:42:55 2010][7894] Finished testing 127.0.0.1.
Time : 8.07 secs

[Mon May 17 21:42:55 2010][7821] user admin : test complete

[Mon May 17 21:42:55 2010][7821] Total time to scan all hosts
: 8 seconds

[Mon May 17 21:42:55 2010][7821] user admin : Kept alive con-
nection

[Mon May 17 21:43:31 2010][22486] scheduler: synscan.nes de-
pends on ping_host.nasl which could not be found, thus this
dependency is not considered for execution sequence

[Mon May 17 21:43:31 2010][22486] scheduler: open-
vas_tcp_scanner.nes depends on ping_host.nasl which could not
be found, thus this dependency is not considered for execu-
tion sequence

[Mon May 17 21:43:31 2010][22486] user admin starts a new
scan. Target(s) : 192.168.1.2, with max_hosts = 20 and
max_checks = 4

[Mon May 17 21:43:31 2010][22486] user admin : testing
192.168.1.2 (::ffff:192.168.1.2) [22552]

[Mon May 17 21:43:55 2010][22552] Finished testing
192.168.1.2. Time : 24.24 secs
```

```
[Mon May 17 21:43:55 2010][22486] user admin : test complete

[Mon May 17 21:43:55 2010][22486] Total time to scan all
hosts : 24 seconds

[Mon May 17 21:43:55 2010][22486] user admin : Kept alive
connection

[Mon May 17 21:49:23 2010][22486] scheduler: synscan.nes de-
pends on ping_host.nasl which could not be found, thus this
dependency is not considered for execution sequence

[Mon May 17 21:49:23 2010][22486]

...

...
```

Der Scan-Log-Datei können Sie eine Fülle an Details zu den geladenen Plug-ins, dem Scan-Zeitpunkt etc. entnehmen. Aufgrund der Fülle an Informationen kann das Scan-Protokoll schnell hundert MB und mehr groß werden. Auch hier ein Blick auf einen Ausschnitt:

```
resolved to name localhost

SSH-DEBUG: Not setting login information for local checks at
127.0.0.1 : No mapping found.

resolved to name 192.168.1.2

SSH-DEBUG: Not setting login information for local checks at
192.168.1.2 : No mapping found.

resolved to name 192.168.1.2

SSH-DEBUG: Not setting login information for local checks at
192.168.1.2 : No mapping found.

resolved to name 192.168.1.4

SSH-DEBUG: Not setting login information for local checks at
192.168.1.4 : No mapping found.

[11811](/usr/local/lib/openvas/plugins/remote-pwcrack-
options.nasl) script_get_preference_file_location: could not
get local file name from preference Passwords file :

resolved to name 192.168.1.2

SSH-DEBUG: Not setting login information for local checks at
192.168.1.2 : No mapping found.

resolved to name 192.168.1.11
```

```
SSH-DEBUG: Not setting login information for local checks at
192.168.1.11 : No mapping found.

[5015](/usr/local/lib/openvas/plugins/remote-pwcrack-
options.nasl) script_get_preference_file_location: could not
get local file name from preference Passwords file :

resolved to name 192.168.1.11

SSH-DEBUG: Not setting login information for local checks at
192.168.1.11 : No mapping found.

[16927](/usr/local/lib/openvas/plugins/remote-pwcrack-
options.nasl) script_get_preference_file_location: could not
get local file name from preference Passwords file :

resolved to name 192.168.1.11

SSH-DEBUG: Not setting login information for local checks at
192.168.1.11 : No mapping found.

[4943](/usr/local/lib/openvas/plugins/remote-pwcrack-
options.nasl) script_get_preference_file_location: could not
get local file name from preference Passwords file :

resolved to name 192.168.1.11

SSH-DEBUG: Not setting login information for local checks at
192.168.1.11 : No mapping found.

[5127](/usr/local/lib/openvas/plugins/remote-pwcrack-
options.nasl) script_get_preference_file_location: could not
get local file name from preference Passwords file :

resolved to name 192.168.1.11

SSH-DEBUG: Not setting login information for local checks at
192.168.1.11 : No mapping found.

[9092](/usr/local/lib/openvas/plugins/remote-pwcrack-
options.nasl) script_get_preference_file_location: could not
get local file name from preference Passwords file :

resolved to name 192.168.1.13

SSH-DEBUG: Not setting login information for local checks at
192.168.1.13 : No mapping found.

[13738](/usr/local/lib/openvas/plugins/remote-pwcrack-
options.nasl) script_get_preference_file_location: could not
get local file name from preference Passwords file :

[15519](/usr/local/lib/openvas/plugins/secpod_open_tcp_ports.
nasl) nasl_keys: bad variable #0 skipped
```

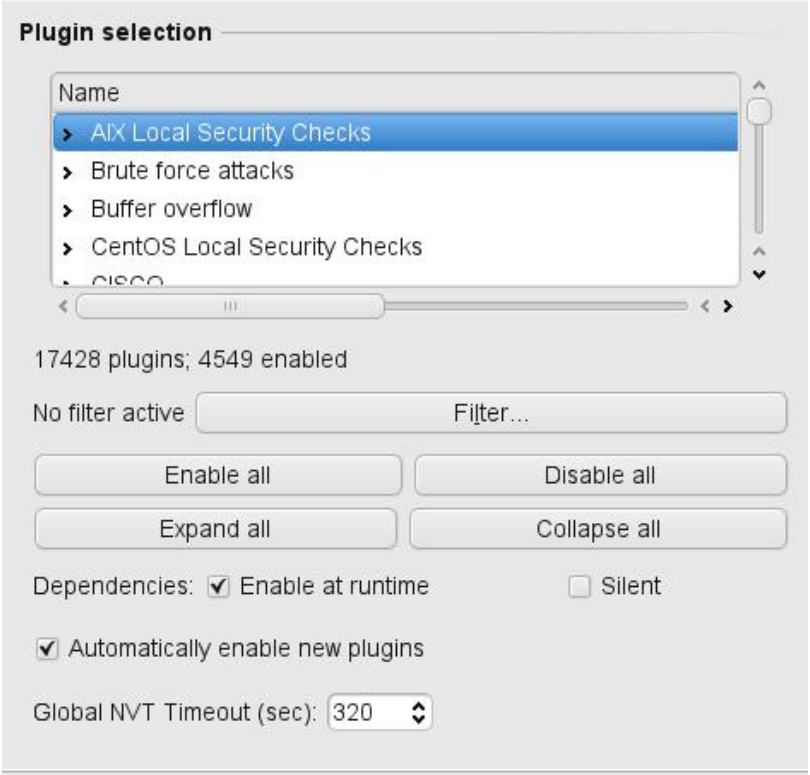
```
[16699](/usr/local/lib/openvas/plugins/GSHB/GSHB_Printer_SSL-
TLS.nasl) nasl_keys: bad variable #0 skipped
```

Unterhalb der verschiedenen allgemeinen Scan-Optionen finden Sie einen Auswahlbereich, über den Sie bestimmen, welche Scanner OpenVAS verwenden soll. Sie haben die Wahl zwischen sechs verschiedenen Typen:

- S**NMP scanner**: OpenVAS verfügt über einen eigenen SNMP-Scanner. Mit diesem können Sie SNMP-Requests an die zu scannenden Systeme verschicken und diese auf mögliche SNMP-Schwachstellen prüfen. Insbesondere Router lassen sich mit diesem Scanner auf Schwachstellen hin überprüfen.
- **TCP scanner**: Dieser Scanner ist auf das Aufdecken von offenen Ports spezialisiert. Er ist standardmäßig aktiviert. Auf dem Register *Advanced* finden Sie verschiedene Konfigurationsmöglichkeiten.
- **Netstat scanner**: Der Netstat-Scanner kommt insbesondere auf Linux-Systemen zum Einsatz. Er zeigt die Protokollstatistiken und aktuellen Rechnernetzverbindungen an. Mit diesen Statistiken finden Sie heraus, welche Ports geöffnet sind oder welche Verbindungen zu entfernten Rechnern bestehen.
- **Ping the remote host**: Auch diese Option ist standardmäßig aktiviert. Sie sorgt dafür, dass vor dem eigentlichen Scan-Vorgang ein Ping an das Zielsystem verschickt wird, um sicherzustellen, dass das System auch erreichbar ist.
- **SYN Scan**: Dieser Scanner sendet ein SYN-Paket an das Ziel und wartet auf die Antwort. Kommt keine, geht OpenVAS davon aus, dass der Port geschlossen ist.
- **Scan for LaBrea tarpitted hosts**: LaBrea (*http://labrea.sourceforge.net*) ist eine sogenannte TCP-Teergrube (Tarpit). Dabei handelt es sich um eine spezielle Honeypot-Form, bei der Netzwerkverbindungen künstlich verlangsamt werden und der Verbindungspartner möglichst lange blockiert wird. Wenn Sie diese Option aktivieren, sucht OpenVAS auch nach den LaBrea-Teergruben.

Außerdem können folgende Port-Scanner aktiviert werden: Portbunny, amap, Pnscan, ike-scan und strobe.

Die Auswahl ist einfach: Aktivieren Sie einfach das Kontrollkästchen rechts neben dem jeweiligen Eintrag.

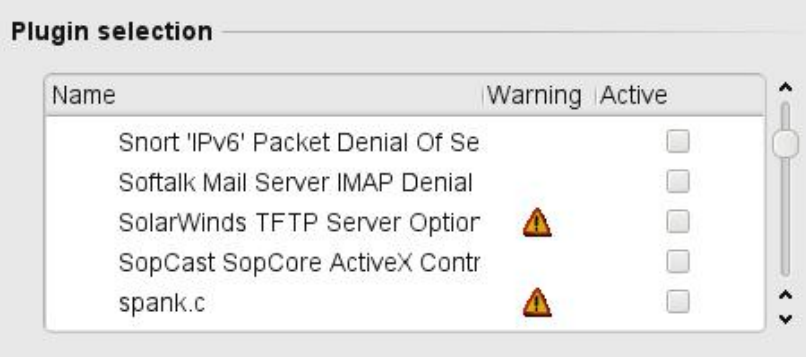


Die Plug-in-Auswahl des OpenVAS-Clients.

3.5.2 Plug-in-Auswahl

Sind die allgemeinen Scan-Einstellungen definiert, so widmet man sich in der Regel im nächsten Schritt der Auswahl der Plug-ins. Wie bereits erwähnt, verbergen sich dahinter die eigentlichen Tests. Die Tests sind in die oben erwähnten Kategorien eingeteilt. Das erleichtert das Auffinden und die Auswahl der gewünschten Tests. Mit einem Häkchen aktivieren Sie einfach die Tests, die OpenVAS an Ihren Zielen durchführen soll.

Sie können auch einzelne Tests einer Testkategorie aktivieren. Dazu öffnen Sie die gewünschte Kategorie, hangeln sich durch die verschiedenen Skripts und aktivieren bzw. deaktivieren die gewünschten NVTs.



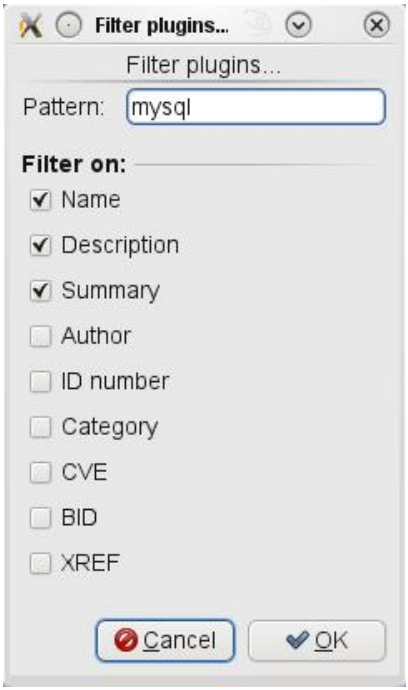
Das Aktivieren einer Testkategorie. Kritische Skripts sind durch das Warnsignal gekennzeichnet.

Neben der Spalte *Active* finden Sie in der Plug-in-Auswahl eine weitere Spalte: *Warning*. Dieses Warnungs-Symbol zeigt an, dass sich durch den Einsatz dieses Plug-ins möglicherweise Dienste auf dem Zielsystem abschalten oder das gesamte System beeinträchtigt werden könnte. Dies wird durch einen speziellen Warnhinweis gekennzeichnet (siehe voranstehende Abbildung).

Bei der Ausführung solcher Testskripts sollten Sie sich also der Gefahr für das Zielsystem bewusst sein. Unter Umständen müssen Sie auf dem Zielsystem beeinträchtigte Dienste manuell neu starten.

Unterhalb der Plug-in-Liste finden Sie die Gesamtzahl der vom Server geladenen Plug-ins. Außerdem zeigt Ihnen der Client an, wie viele Plug-ins von Ihnen aktiviert wurden.

Eine weitere Besonderheit der Plug-in-Auswahl: Sie können über die Filterfunktion die für Sie interessanten Skripts ausfindig machen und aktivieren. Um Plug-ins einzusetzen, die bestimmten Kriterien entsprechen, klicken Sie auf die Schaltfläche *Filter* und geben Sie in das Eingabefeld *Pattern* Ihr Suchmuster ein.



Die Filterfunktion der Plug-in-Auswahl erlaubt die gezielte Auswahl der Test-Skripts.

Sie können die Suche auf folgende Bereiche einschränken:

- Name
- Beschreibung
- Zusammenfassung
- Autor
- Skript-ID
- Kategorie
- CVE
- BID
- XREF

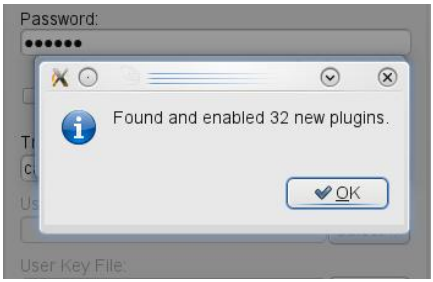
Beachten Sie bei der Filterung, dass Ihre vorherige Plug-in-Auswahl komplett gelöscht wird, sobald Sie einen Filter aktivieren.

Unterhalb der Filter-Schaltfläche finden Sie vier weitere Buttons:

- **Enable all**: Ein Klick schaltet alle Plug-in-Kategorien ein.
- **Disable all**: Schaltet alle Plug-in-Kategorien aus.
- **Expand all**: Ein Klick klappt die Plug-in-Liste soweit auf, dass sämtliche Plugins angezeigt werden.
- **Collapse all**: Mit einem Klick werden nur noch die Plug-in-Kategorien angezeigt.

Es folgen zwei Einstellungen zu den Abhängigkeiten. Wenn Sie die Option *Enable at runtime* aktivieren, schaltet der OpenVAS-Server alle Plug-ins ein, die von den bereits selektierten abhängig sind. Wenn Sie die Option *Silent* aktivieren, sendet der Server keinen Bericht für die Plug-ins, die nicht explizit eingeschaltet wurden.

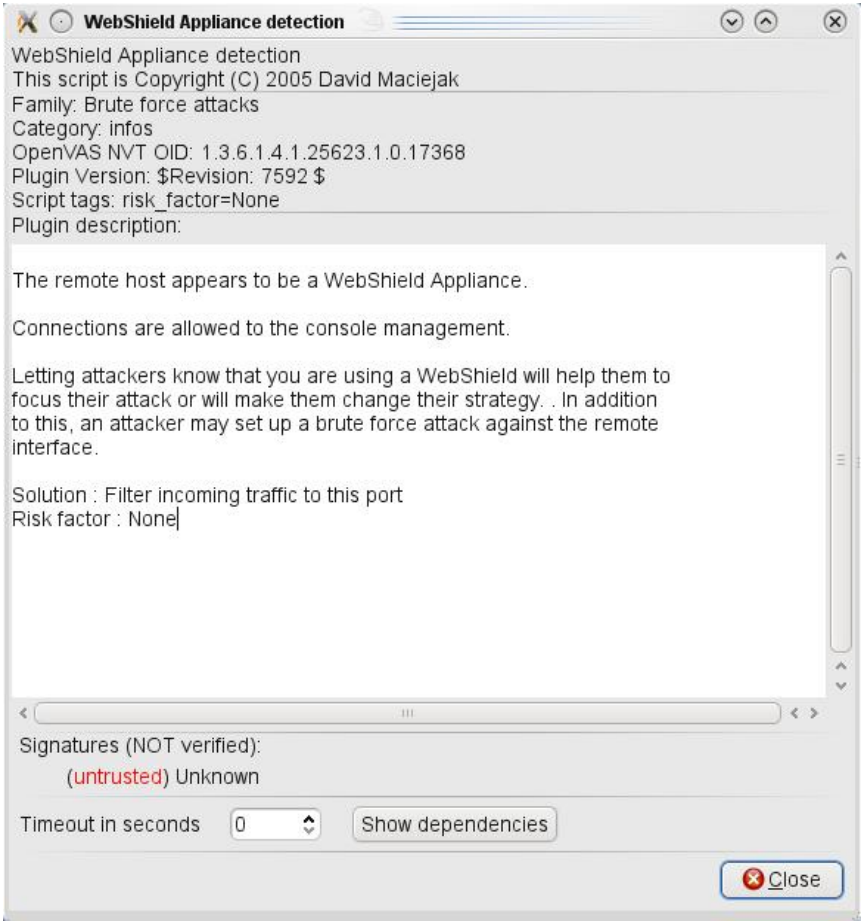
Standardmäßig ist außerdem die Option *Automatically enable new plugins* aktiviert. Wie Sie bereits wissen, kann OpenVAS seine Plug-ins automatisch auf den neusten Stand bringen. Durch das Aktivieren dieser Option stellen Sie sicher, dass immer auch die neu hinzugekommenen aktiviert werden.



Der Clients hat 32 neue Plug-ins gefunden und aktiviert.

Nachdem der Client eine Verbindung zum OpenVAS-Server aufgebaut hat und die Plug-ins geladen wurden, gibt der Client einen Hinweisdialog aus, dem Sie entnehmen können, wie viele neuen Plug-ins gefunden und ob diese aktiviert wurden.

Mit der Option *Global NVT Timeout* bestimmen Sie außerdem, wie lange auf eine Antwort des NVT Feed Service für die Überprüfung der lokalen Plug-in-Datenbank gewartet wird.



Die Details zu einem Plug-in.

Das OpenVAS-Team gibt sich zwar bei der Vergabe der Plug-in-Bezeichnungen alle Mühe, diesen eine aussagekräftige Bezeichnung zuzuweisen, doch was, wenn Sie mehr über ein Plug-in wissen wollen?

Klicken Sie doch einfach doppelt auf einen Plug-in-Eintrag. Es öffnet sich der Plug-in-Info-Dialog, dem Sie eine Fülle an Detailinformationen entnehmen können.

Neben der Plug-in-Bezeichnung und der Copyright-Info zeigt der Info-Dialog im Kopfbereich folgende Informationen an:

- Plug-in-Familie
- Plug-in-Kategorie
- OpenVAS NVT ID
- Plug-in-Version

Im Bereich *Plugin description* finden Sie die Beschreibung, die der Plug-in-Entwickler in den Test-Skripts hinterlegt hat. Da alle angezeigten Informationen dem Plug-in selbst entnommen sind, sind auch Art und Inhalt der Beschreibung sehr unterschiedlich bis nicht vorhanden.

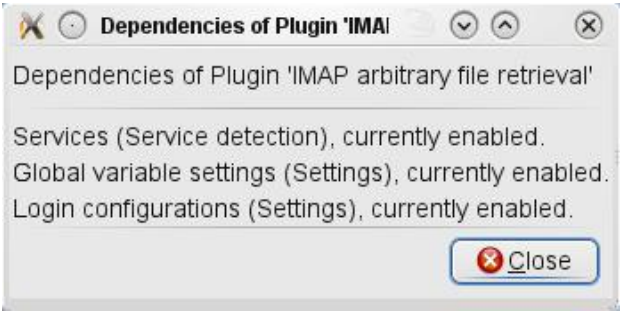
Neben einer allgemeinen Problembeschreibung finden Sie in der Beschreibung meist auch einen Verweis zur relevanten Security Advisory, eine Kurzinfo zur Problemlösung und eine Bewertung des Risikofaktors. Hier ein Beispiel, das ein IMAP-Server-Plug-in beschreibt:

```
The target is running an IMAP daemon that allows an authenti-
cated user to retrieve and manipulate files that would be
available to that user via a shell.  If IMAP users are denied
shell access, you may consider this a vulnerability.


See also : http://www.washington.edu/imap/IMAP-
FAQs/index.html#5.1

Solution : Contact your vendor for a fix.

Risk factor : Medium
```

Der Info-Dialog bietet Ihnen nicht nur interessante und nützliche Hintergrundinformationen, sondern noch mehr. So können Sie unter *Signatures* die dem Server bekannten Zertifikate anzeigen. Wenn das im Info-Dialog angezeigte Plug-in einfach oder mehrfach signiert wurde, werden hier Name und Vertrauensgrad der zu den Signaturen gehörenden Zertifikate aufgeführt. Bei signierten Plug-ins können Sie auch die Zertifikate in Augenschein nehmen.

Mit dem Eingabefeld *Timeout in seconds* geben Sie die Zeitspanne an, nach der die Ausführung des Plug-ins abgebrochen wird, falls es seine Ausführung noch nicht abgeschlossen hat.

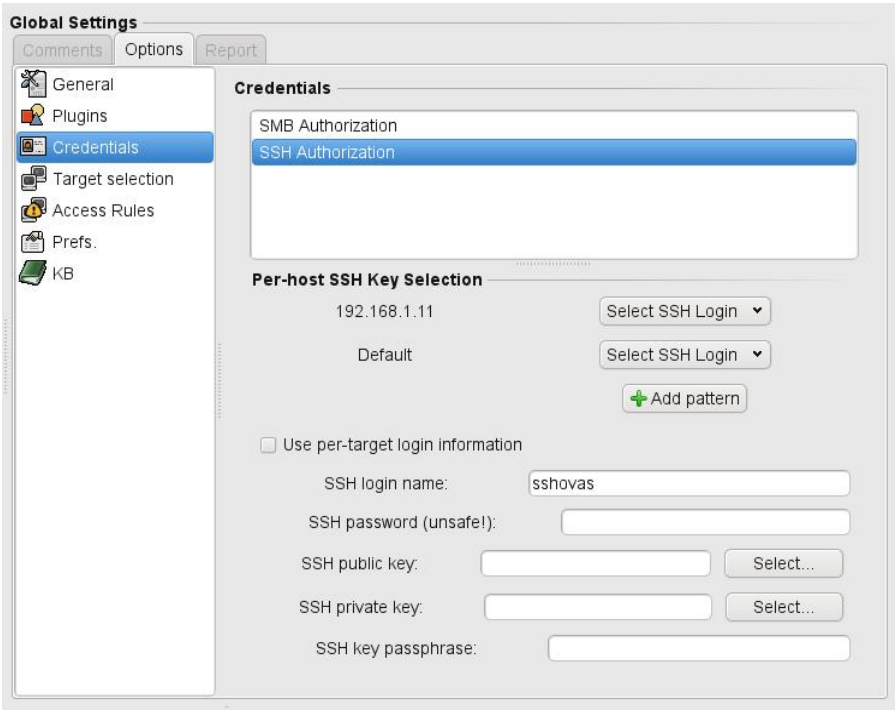
Mit einem Klick auf die Schaltfläche *Show dependencies* wird eine Liste der Plug-in-Abhängigkeiten geöffnet. Dabei wird auch der Status (eingeschaltet/ausgeschaltet) der Plug-ins angezeigt, von denen das aktuelle Plug-in abhängt.



Die Abhängigkeiten eines Plug-ins.

3.5.3 Systemspezifische Schwächen identifizieren

OpenVAS unterschied sich von Anfang an von anderen Produkten dadurch, dass der Scanner nicht nur typische Netzwerkschwachstellen erkennen, sondern auch systemspezifische Sicherheitslücken identifizieren kann. Die dafür relevanten Funktionen finden Sie auf dem Register *Credentials.*



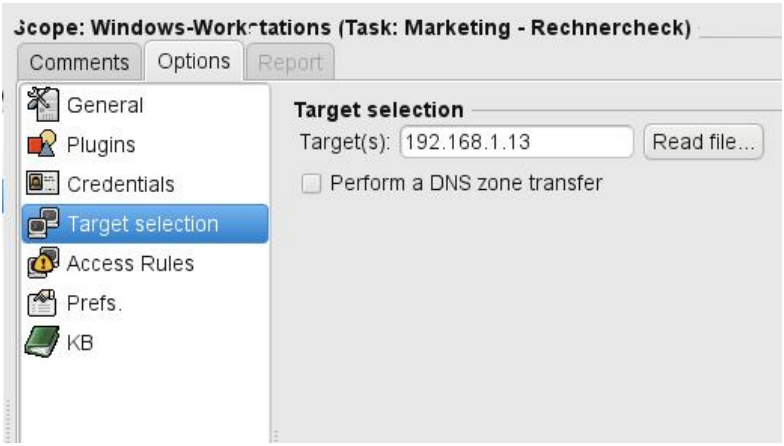
Die Funktionen des Registers *Credentials* erlauben das Aufdecken von systemspezifischen Schwachstellen.

Das *Credentials*-Register stellt Ihnen ein Auswahlmenü zur Verfügung, über das Sie aus drei Bereichen Einstellungen wählen können. Jede Option stellt Ihnen eigene Konfigurationsmöglichkeiten zur Verfügung:

- **SMB Authorization**: Erlaubt Ihnen das Testen von mehreren Windows-Domains. Sie können bis zu vier Domains prüfen.

- **SSH Authorization**: Erlaubt es Ihnen, sich über SSH Zugang zu Unix-basierten Systemen zu verschaffen.

Auf die Details kommen wir im nächsten Kapitel zu sprechen.



Die Zielauswahl.

3.5.4 Die Zielauswahl

Neben den Plug-ins, die Sie auf das bzw. die Ziele anwenden, ist die wichtigste Konfiguration natürlich die Zielauswahl. Hierfür verwenden Sie das Register *Target selection*. Wie Sie voranstehender Abbildung entnehmen können, ist diese sehr übersichtlich.

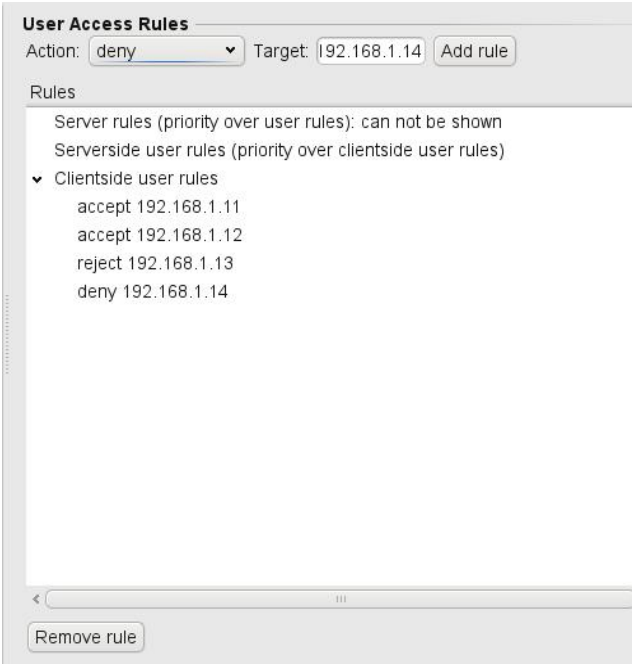
In das Eingabefeld *Target(s)* geben Sie den bzw. die Hosts ein, die OpenVAS überprüfen soll. Wenn Sie mehrere Hosts testen wollen, können Sie entweder einen IP-Adressbereich oder eine kommaseparierte Liste der Hostnamen verwenden.

Auch die Kombination aus Hostnamen und Adressbereich ist möglich. Wichtig dabei: Verwenden Sie kein Leerzeichen zwischen den Namen bzw. IP-Adressen. Ein Beispiel:

```
hostname1,hostname2,168.192.1.11
```

Sie können die zu scannenden Systeme auch in einer Textdatei hinterlegen und auf diese Datei über den Button *Read File* zugreifen. Der Vorteil ist offensichtlich: Sie müssen die Ziele nicht manuell in das Target(s)-Feld eingeben, sondern können diese in der Zielliste nach Belieben verwalten. Die Zielliste kann eine oder mehrere Zeilen mit kommaseparierten Einträgen enthalten.

Wenn Sie die Option *Perform a DNS zone transfer* aktivieren, führt der OpenVAS AXFR-Anfragen (Zonen-Transfer) an den Nameserver des Ziels durch, um damit die Liste der Hosts der Ziel-Domäne zu ermitteln.



Die Konfiguration der Zugriffssteuerung.

3.5.5 Zugriffssteuerung

Der OpenVAS-Client erlaubt auch das Anlegen und Verwalten von Zugriffsregeln. Damit bestimmen Sie, welche Zielrechner gescannt werden dürfen und welche nicht. So können Sie gezielt Systeme von den Sicherheitstests ausschließen. In dem Eingabefeld können einzelne Systeme und auch Subnetze angegeben werden.

OpenVAS kennt drei verschiedene Regeltypen:

- **Serverregeln**: Die Regeln dieses Typs gelten für den gesamten Server und betreffen alle Benutzer, die sich mit diesem Server verbinden.
- **Serverseitige Benutzerregeln**: Diese Regeln gelten nur für einen bestimmten Benutzer und nur den Nutzer, unabhängig davon, von welchem Client aus er sich mit dem Server verbindet.
- **Clientseitige Benutzerregeln**: Dieser Typ gilt für den jeweiligen Client. Diese Regeln betreffen nur den Bereich, in dem sie definiert sind.

Dabei gibt es eine klare Hierarchie: Serverregeln haben eine höhere Priorität als serverseitige Regeln, die wieder eine höhere Priorität als die clientseitigen Regeln besitzen.

Sie finden die Serverregel standardmäßig in folgendem Verzeichnis:

```
/etc/local/openvas/openvasd.rules
```

Die Zugriffsregeln wirken selbstbeschränkend. Serverseitige Regeln werden nur als Informationen für den Client angeboten. Diese Regelsätze greifen unabhängig voneinander.

Syntax:

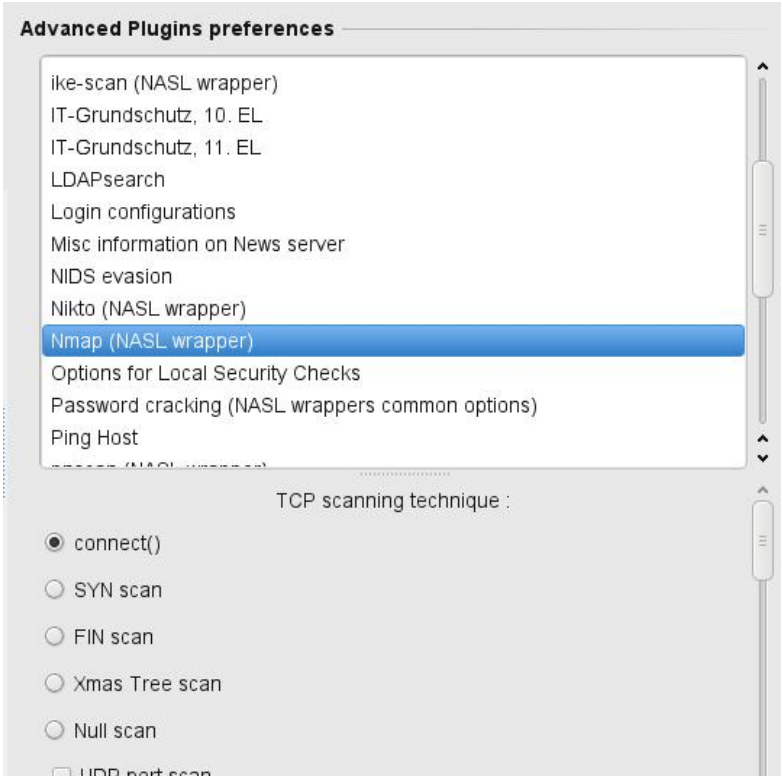
```
SERVER <|> RULES <|>
regel_1;
regel_2;
regel_3;
...
<|> SERVER
CLIENT <|> RULES <|>
regel_1;
regel_2;
regel_3;
...
<|> CLIENT
```

Die Server- und die serverseitigen Regelsätze übermittelt der Server nur zu Informationszwecken an den Client. Diese Regelsätze werden aufseiten des OpenVAS-Servers verwaltet und können nicht vom Client verändert werden. Mithilfe des OpenVAS-Clients können Sie nur einen Client-seitigen Regelsatz erstellen und bearbeiten.

Wie sieht nun eine Client-Regel aus und wie legt man sie an? Eine Regel besteht aus einer Aktion und einem Ziel. Das Ziel geben Sie in dem *Target*-Eingabefeld an, die Aktion wählen Sie aus Auswahlmenü *Action*. Sie haben die Wahl zwischen folgenden Aktionen:

- accept
- deny
- reject

Sie sichern die Regel mit einem Klick auf die Schaltfläche *Add rule*. Sie wird anschließend in der Regelliste aufgeführt. Dort kann sie durch Markieren und einen anschließenden Klick auf die *Remote rule*-Schaltfläche auch wieder entfernt werden.



Ein Blick auf die erweiterten Plug-in-Einstellungen.

3.5.6 Die erweiterten Plug-in-Einstellungen

Der vorletzte Register *Prefs.* dient der Konfiguration der erweiterten Plug-in-Einstellungen. Nach der Wahl eines Plug-ins bzw. eines Zusatzmoduls werden die verfügbaren Einstellungen unterhalb der Auswahl aufgeführt.

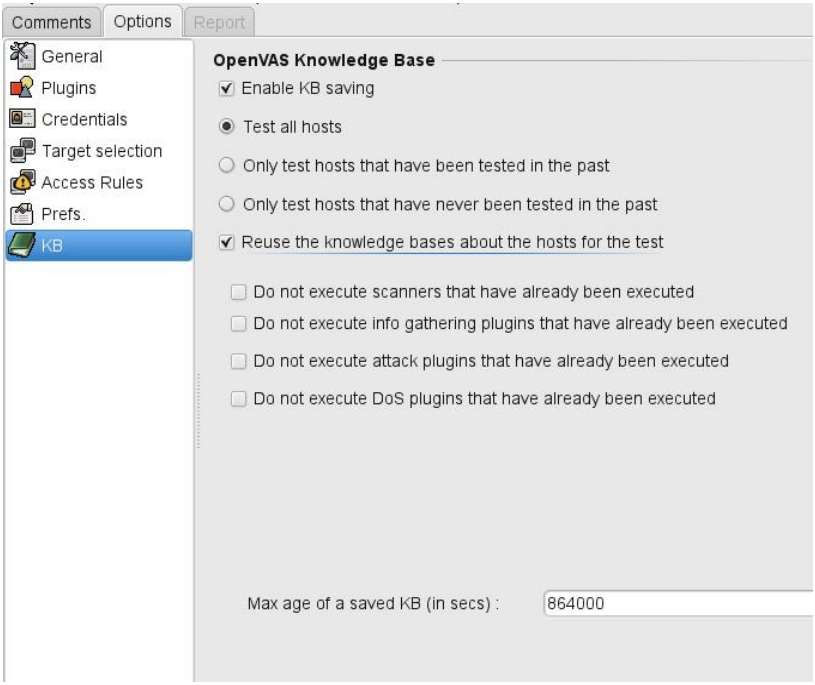
Hier können Sie beispielsweise verschiedene Account-Informationen für das Testen von Web-Services oder auch Einstellungen für die Ausführung der Port-Scanner hinterlegen.

Auf die Details gehen wir in Kapitel 4 ein.

3.5.7 Die Wissensbasis

In der sogenannten Wissensbasis (Knowledge Base oder kurz KB) kann OpenVAS die serverseitigen Scanergebnisse verwalten. Dabei handelt es sich um Informationen, die während des Scans eines Zielrechners gesammelt und dann in einer Wissensbasis zusammengetragen werden. OpenVAS legt pro Zielrechner eine Wissensbasis an.

Die Knowledge Base muss zunächst auf dem Register *KB* aktiviert und gegebenenfalls an Ihre Anforderungen angepasst werden. Bei einer Standardkonfiguration wird die jeweilige Wissensbasis automatisch gelöscht, sobald der Scan des Zielrechners abgeschlossen ist. Allerdings kann es auch Sinn machen, diese Daten zu bewahren, um zu einem späteren Zeitpunkt auf sie zurückgreifen zu können.



Die Einstellungen der Knowledge Base.

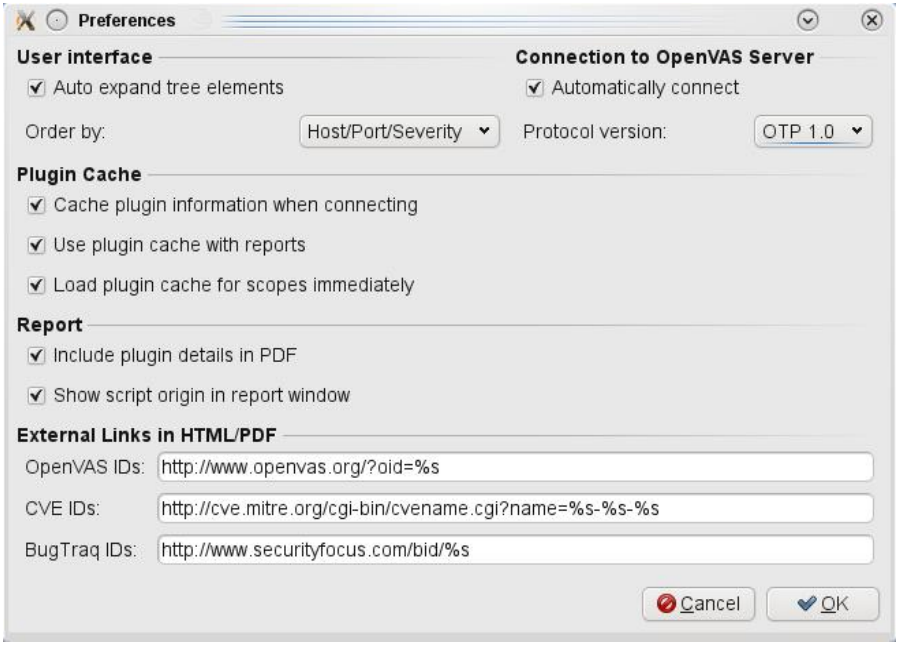
Auch auf die Knowledge Base kommen wir in Kapitel 4 noch detailliert zu sprechen.

3.6 Die Programmeinstellungen des OpenVAS-Clients

Über das Menü *File> Preferences* können Sie verschiedene programmübergreifende Einstellungen des OpenVAS-Clients bearbeiten. Dazu gehören beispielsweise Einstellungen, die die Benutzerschnittstelle, den Plug-in-Cache und die Berichte betreffen.

Im Bereich *User interface* legen Sie zunächst mit der Option *Auto expand tree elements* fest, ob die Baumstruktur automatisch aufgeklappt wird. Ist das der Fall, können Sie die Reihenfolge ändern. Sie haben die Wahl zwischen zwei Optionen:

- Host/Port/Severity
- Port/Host/Severity



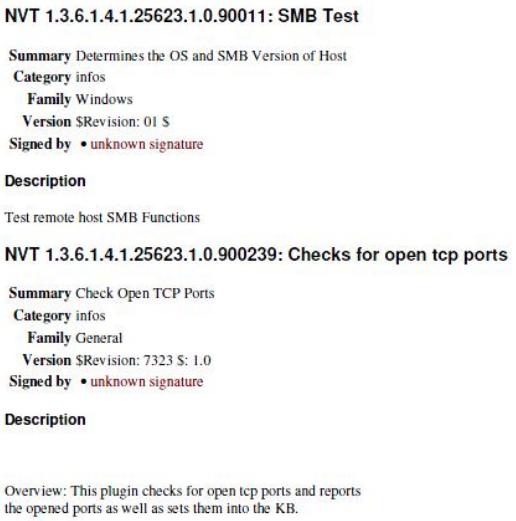
Die Programmeinstellungen des OpenVAS-Clients.

Im Bereich *Connection to OpenVAS Server* können Sie die automatische Verbindungsaufnahme aktivieren. Außerdem scheint die Auswahl der Protokollversion

vorgesehen zu sein. Beim OpenVAS-Client 3.0.0 steht allerdings nur die Protokollversion OTP 1.0 zur Verfügung.

Es folgen drei Einstellungen zum Plug-in-Cache:

- **Cache plugin information when connecting**: Diese Option ist wie die beiden folgenden, standardmäßig aktiviert und zeigt die Cache-Informationen bei der Verbindungsaufnahme zwischen Client und Server an.
- **Load plugin cache for scopes immediately**: Sorgt für ein unmittelbares Laden des Plug-in-Cache für die Bereiche.



NVT 1.3.6.1.4.1.25623.1.0.90011: SMB Test

Summary Determines the OS and SMB Version of Host
Category infos
Family Windows
Version $Revision: 01 $
Signed by • unknown signature

Description

Test remote host SMB Functions

NVT 1.3.6.1.4.1.25623.1.0.900239: Checks for open tcp ports

Summary Check Open TCP Ports
Category infos
Family General
Version $Revision: 7323 $: 1.0
Signed by • unknown signature

Description

Overview: This plugin checks for open tcp ports and reports
the opened ports as well as sets them into the KB.

Die Plug-in-Details in einem PDF-Bericht.

Als Nächstes können Sie zwei Berichtfunktionen anpassen:

- **Include plugin details in PDF**: Standardmäßig werden die Plug-in-Details in einen PDF-Bericht exportiert.
- **Show script origin in report window**: Der Ursprung eines Skripts wird ebenfalls bei einer OpenVAS-Standardinstallation im Bericht angezeigt.

Den Abschluss der Programmeinstellungen des OpenVAS-Clients bildet die Link-Konfiguration. Wenn Sie Ihre Berichte nach HTML oder PDF exportieren, werden standardmäßig die Web-Links zu den OpenVAS-, CVE- und BugTraq-IDs eingefügt.

3.7 Der Windows-Client

Neben dem Linux-Client gibt es für OpenVAS auch einen Windows-basierten Client. Er trägt die Bezeichnung GSA Desktop. Er befand sich zum Zeitpunkt, als dieses Buch entstand, noch in einer sehr frühen Entwicklungsphase.

Er wird die gleiche Funktionalität wie die Linux-Variante bieten. Seine Besonderheit: Er wird auch das neue Protokoll OMP unterstützen (siehe Anhang C).

4 Die Scan-Einstellungen im Detail

Nachdem Sie nun einen Überblick, in Teilbereichen sogar einen vollständigen Überblick darüber haben, welche Einstellungen die verschiedenen Register zu bieten haben, schauen wir uns als Nächstes die Einstellungen an, die wir bislang nicht eingehend beschrieben haben. Insbesondere die Funktionen der Register *Credentials*, *Prefs* und *KB* sind bislang nicht ausreichend besprochen werden.

4.1 Credentials-Einstellungen

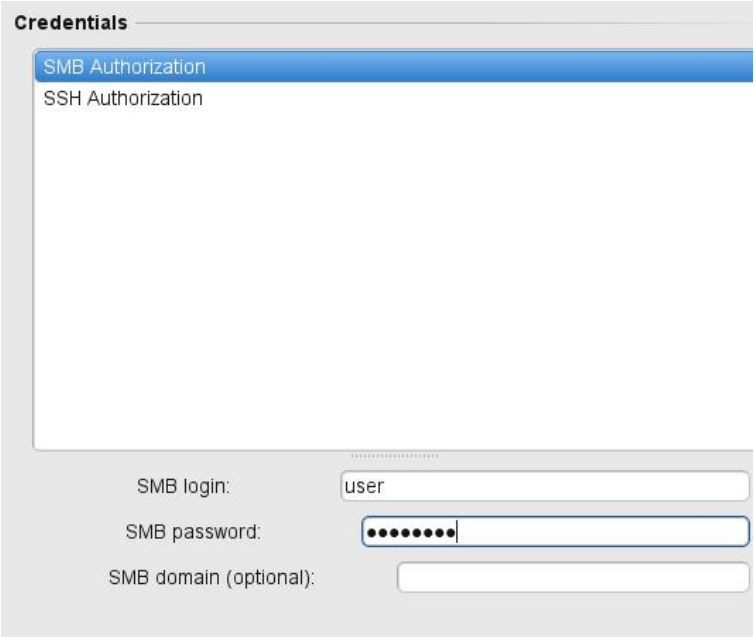
Wie bereits erwähnt, ist eine Besonderheit von OpenVAS, dass der Security-Scanner nicht nur Systeme von außen, sondern auch von innen prüfen kann. Die dafür relevanten Funktionen finden Sie auf dem Register *Credentials*. Hier können Sie die Zugänge zu zwei Diensten konfigurieren:

- SMB
- SSH

Auf die Einstellungen für den jeweiligen Bereich greifen Sie über das Auswahlmenü im oberen Bereich des Registers zu. Der erste Eintrag trägt die Bezeichnung *SMB Authorization* und erlaubt den Zugriff auf Windows-basierte Netzwerkfreigaben, die auf dem SMB-Protokoll von Microsoft beruhen.

SMB steht für Server Message Block und ist auch unter den Bezeichnungen LAN-Manager- oder NetBIOS-Protokoll bekannt. Es ist ein weitverbreitetes Kommunikationsprotokoll für Datei-, Druck- und andere Serverdienste in Netzwerken. Es ist der Kern der Netzwerkdienste der Windows-Produktfamilie.

Außerdem wird es in der frei verfügbaren Software Samba und Samba-TNG verwendet, um Windows-Systemen den Zugriff auf Ressourcen von UNIX-basierten Systemen zu ermöglichen und umgekehrt.



Die Konfiguration des SMB-Zugriffs.

Mit den Credentials-Einstellungen kann sich Ihre OpenVAS-Installation also Zugriff auf Windows-Freigaben verschaffen – auch auf Samba-basierte Dienste.

Der Vorteil ist offensichtlich: Der Security-Scanner kann einen Blick ins Innenleben der Systeme werfen, und Sie können auf diesem Weg weit mehr potenzielle Sicherheitslücken identifizieren. Sie können so auch prüfen, ob auf dem jeweiligen Windows-System die aktuellsten Patches aufgespielt sind. Derlei Tests sind bei „normalen“ Sicherheitsscans nicht möglich.

Abhängig von der Konfiguration und den Rechten des verwendeten SMB-Accounts können Sie mit dieser Funktion sogar ganze bzw. mehrere Windows-Domains unter die Lupe nehmen. Auf das Scannen von Windows-Systemen und die notwendigen Änderungen aufseiten der zu analysierenden Systeme kommen wir später noch zu sprechen.

Die Windows-Credentials-Funktionen sind insbesondere für die Analyse von Windows NT, 2000, XP, 2003 Server und Vista geeignet.

Auf dem zugehörigen Formular können Sie einen SMB-Account einrichten. Dazu können folgende Angaben hinterlegt werden, wobei die letzte Angabe optional ist:

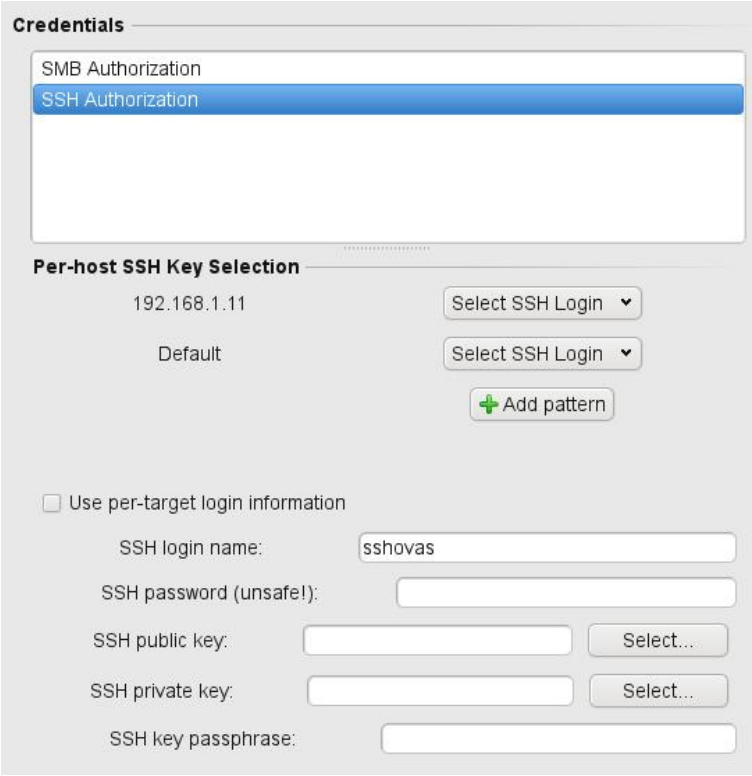
- SMB-Account
- SMB-Passwort
- SMB-Domain

Die zweite Zugangsmöglichkeit, die das Register *Credentials* bietet, ist der SSH-basierte Zugang. SSH oder auch Secure Shell ist ein gängiges Netzwerkprotokoll, mit dem bzw. mit dessen Implementierungen man auf eine sichere Art und Weise eine verschlüsselte Netzwerkverbindung mit einem entfernten Computer herstellt.

Es wird häufig verwendet, um sich eine entfernte Kommandozeile quasi auf den lokalen Rechner zu holen, das heißt, auf der lokalen Konsole werden die Ausgaben der entfernten Konsole angezeigt und die lokalen Tastatureingaben werden an den entfernten Rechner gesendet. Für den Benutzer ist es, als säße er direkt vor dem entfernten System.

Der SSH-Client – in unserem Fall übernimmt diese Rolle der OpenVAS-Server, der vom OpenVAS-Client gesteuert wird – kann sich wahlweise per Public-Key-Authentifizierung mit einem privaten Schlüssel, dessen öffentlicher Schlüssel auf dem Server hinterlegt ist, oder einem gewöhnlichen Kennwort authentifizieren.

Während Letzteres immer eine Benutzerinteraktion erfordert (solange das Kennwort nicht unverschlüsselt auf dem Client-Rechner gespeichert werden soll), ermöglicht die Public-Key-Authentifizierung, dass sich Client-Computer auch ohne Benutzerinteraktion auf SSH-Servern einloggen können, ohne dass dabei ein Passwort auf dem Client im Klartext gespeichert werden muss.



Die Einstellungen für die SSH-Autorisierung.

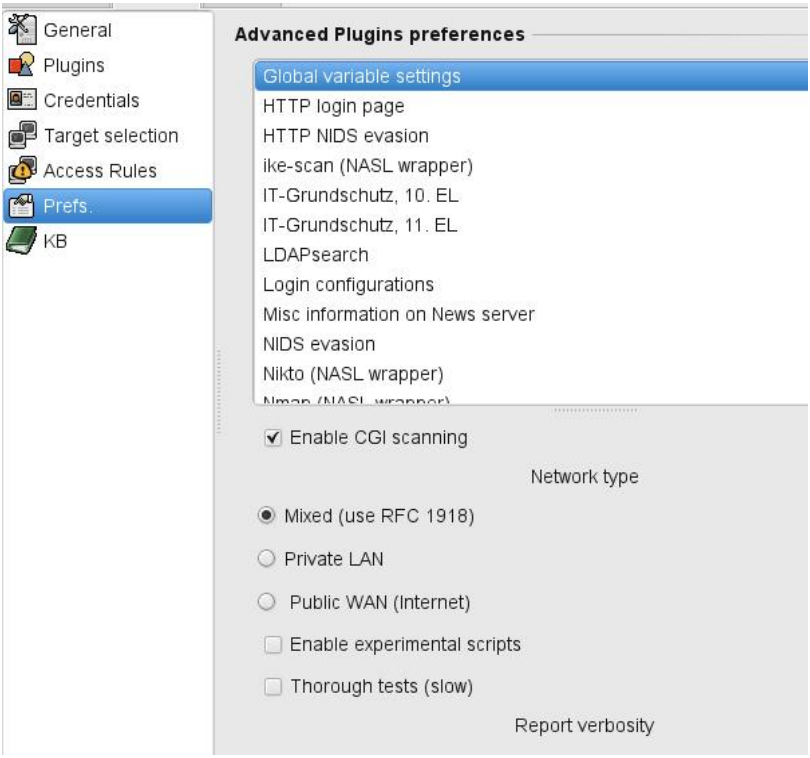
Die SSH-Einstellungen unterstützen die verschiedenen Authentifizierungsmöglichkeiten. Diese können in dem Formular folgende Daten hinterlegen:

- SSH-Benutzername
- SSH-Passwort
- öffentlicher Schlüssel für die SSH-Verwendung
- privater Schlüssel für die SSH-Verwendung
- Passwortphrase für den SSH-Schlüssel

Die beiden Schlüssel können Sie über die Schaltflächen *Select* bequem auswählen.

Die SSH-Funktionen eignen sich naturgemäß insbesondere für die Analyse von Unix-/Linux-basierten Systemen, da gerade OpenSSH auf vielen kritischen Systemen zum Einsatz kommt.

Leider ist im OpenVAS-Client die Möglichkeit verschwunden, sich über die Option *Cleartext protocol settings* Zugriff auf Dienste wie Telnet oder RSH zu verschaffen. Diese Möglichkeit war im Vorläufer Nessus 2.x gegeben.



Auf dem Register *Prefs.* geht es noch einmal richtig zur Sache.

4.2 Die erweiterten Einstellungen

Die erweiterten Plug-in-Einstellungen finden Sie auf dem Register *Prefs*. Hier finden Sie eine Liste, über die Sie die verschiedenen Konfigurationsbereiche auswählen. Bevor wir uns die einzelnen Konfigurationsmöglichkeiten genauer ansehen, hier die Bereiche im Überblick:

- 3com switch2hub
- amap
- Availability of scanner helper tools
- Compliance Tests
- CPE-base Policy Check
- Global variable settings
- HTTP NDIS Evasion
- HTTP login page
- ike-scan
- IT-Grundschutz, 10 EL.
- IT-Grundschutz, 11 EL.
- LDAPsearch
- Login configurations
- Misc information on News server
- NIDS evasion
- Nikto
- Nmap
- Options for Local Security Checks
- Password cracking
- Ping host
- pnscan
- portbunny
- Search in LDAP
- Services
- SLAD Run
- SMTP settings
- SNMP settings

- snmpwalk
- SSL Cipher Settings
- strobe
- w3af
- wapiti
- Web mirroring

Beachten Sie, dass die nachfolgenden Beschreibungen auf dem OpenVAS-Client 3.0.0 basieren. Die verfügbaren Funktionen können von Version zu Version variieren – auch bei kurz aufeinanderfolgenden Versionen.

Hinweis

Wie wir später noch sehen werden, können Sie aber auch die Optionen und Funktionen nutzen, die nicht über einen Client verfügbar sind. Dazu müssen Sie lediglich die Scan-Einstellungen editieren und dann die entsprechenden Konfigurationen vornehmen.

4.2.1 3com switch2hub

Ist ein Host mittels eines Switches an das Netzwerk angebunden, so kann er in einen Hub-Zustand gebracht werden, wenn man ihn mit sehr sehr vielen MAC-Adressen überflutet. Mit „mehr“ sind mehr als 1 Millionen Pakete gemeint, wobei bei diesem Angriffstyp zufällige MAC-Adressen verwendet werden.

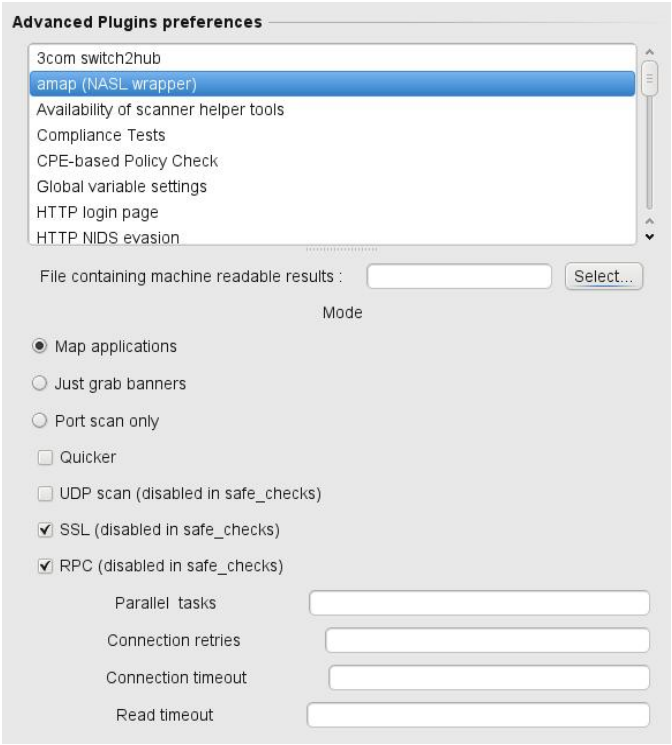
Bei einer solchen Attacke schaltet ein Switch in den Learning-Modus und der Traffic kann ungehindert passieren.

Ein Angreifer kann diesen Angriffstyp nutzen, um Daten zu sniffen, die an das Zielsystem übertragen werden.

Dieser Test-/Angriffstyp weist ein hohes Risiko auf. Schützen kann man sich indes recht einfach: Man sperrt die MAC-Adressen für jeden Port oder man wechselt zu einem neuen Geräte.

Für die Konfiguration dieses Moduls stehen Ihnen drei Einstellungen zur Verfügung:

Option	Beschreibung
Network interface on OpenVAS box	Geben Sie hier die Netzwerkschnittstelle an, die von OpenVAS verwendet wird.
Fake IP	In diesem Eingabefeld hinterlegen Sie die gefälschte IP-Adresse.
Number of packets	Auch die Anzahl der Pakete kann angepasst werden. Sie sollten keinen Wert kleiner als 1.000.000 verwenden.



Die amap-Konfiguration.

4.2.2 amap

Neben nmap ist amap (*http://freeworld.thc.org/thc-amap/*) der zweite wichtige Portscanner, der in OpenVAS integriert ist. Die Stärke von amap: Sie identifizieren

damit Services auf dem bzw. den Zielsystemen, auch dann, wenn diese nicht Standard-Ports verwenden. Bei ssh wird beispielsweise häufig aus Sicherheitsgründen der Standardport 22 auf einen alternativen Port gelegt.

Option	Beschreibung
File containing machine readable results	Erlaubt die Verwendung einer Datei, in der die zu analysierenden Hosts und Ports spezifiziert sind. Wichtig dabei: Die Datei muss von nmap erzeugt worden sein, und zwar mit der Option *-oM*. Ob in der Liste TCP- und UDP-Ports gemischt sind, ist amap gleich.
Map application	Das ist der Standardmodus, in dem amap ausgeführt wird. Dabei werden Trigger gesendet und die Antworten analysiert.
Just grab Banners	Stellt ein Client eine Verbindung zu einem Dienst her, erhält er eine Banner Grab, eine lesbare Zeichenfolge. Anhand des Typs der erhaltenen Zeichenfolge werden in der Regel die Betriebssysteme und die Server-Typen identifiziert. Bei diesem Ausführungsmodus interessiert sich amap nur für die Banner Grab.
Port Scan only	In diesem Modus agiert amap als (fast) reiner Portscanner.
Quicker	Wenn Sie dieses Kontrollkästchen aktivieren, werden Trigger an den Port nur bis zur ersten Identifikation gesendet. Das führt zu einem spürbaren Performancegewinn.
UDP Scan only	Führt lediglich UDP-Scans durch. Diese Option ist deaktiviert, wenn Sie OpenVAS mit der *Safe Check*-Option ausführen.
SSL	Verwendet SSL für die Verbindungsabsicherung. Auch diese Option ist deaktiviert, wenn Sie OpenVAS mit der *Safe Check*-Option ausführen.
RPC	Verwendet lediglich Remote Protocol Calls. Diese Option ist deaktiviert, wenn Sie OpenVAS mit der *Safe Check*-Option ausführen.
Parallel tasks	Hier geben Sie an, wie viele parallele Verbindungen amap nutzt. Der Standardwert ist 32, der Maximalwert 256.
Connection retries	Wenn es bei einem TCP-Connect zu einem Timeout kommt, versucht amap erneut, eine Verbindung zum

	Zielsystem aufzubauen. Hier bestimmen Sie, wie viele Versuche amap unternimmt. Der Standardwert ist 3.
Connection timeout	In diesem Eingabefeld bestimmen Sie das Timeout-Intervall für den Verbindungsaufbau.
Read timeout	Hier den Timeout-Wert für das Lesen.

4.2.3 Availability of scanner helper tools

Mit den beiden Einstellungen dieses Bereich legen Sie fest, ob OpenVAS die Verfügbarkeit der Hilfsprogramme wie amap & Co prüft.

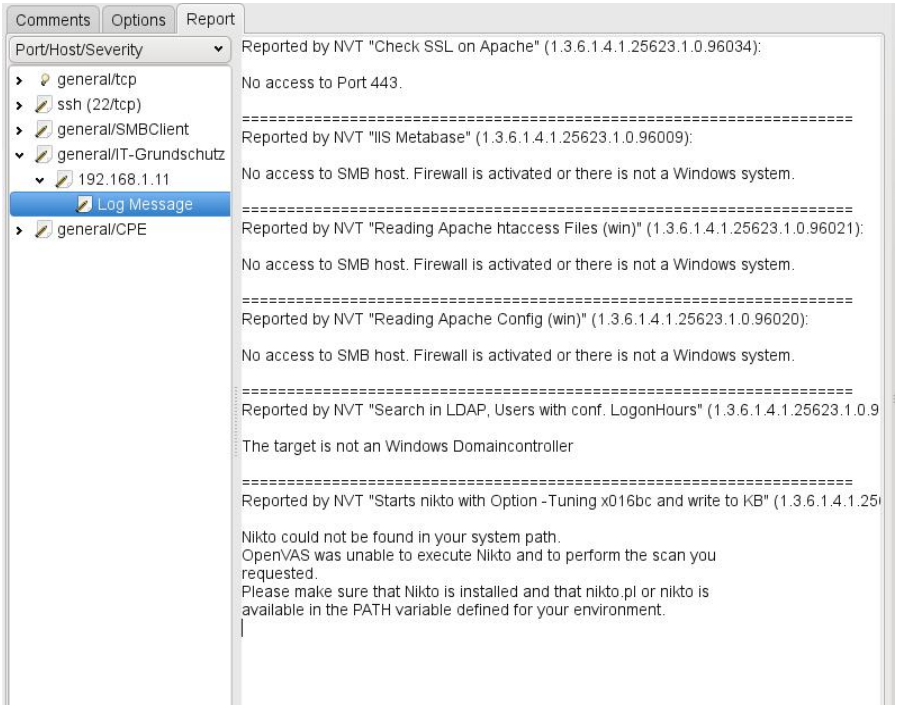
4.2.4 Compliance Tests

In diesem Bereich bestimmen Sie, welche Nachweisprüfungen von OpenVAS durchgeführt werden. OpenVAS unterstützt IT-Grundschutz 10 EL und 11 EL. Als IT-Grundschutz werden die Standardsicherheitsmaßnahmen für typische IT-Objekte (Anwendungen, IT-Systeme, Räumlichkeiten, Netze) bezeichnet.

Das IT-Grundschutzkonzept verzichtet auf eine detaillierte Risikoanalyse, sondern geht vielmehr von pauschalen Gefährdungen aus und definiert personelle, technische, organisatorische und infrastrukturelle Sicherheitsmaßnahmen. Für weitere Informationen sei auf die Website des Bundesamts für Sicherheit in der Informationstechnik (BSI, *http://www.bsi.bund.de*) verwiesen.

Im OpenVAS-Client können Sie drei Einstellungen zur Nachweisprüfung vornehmen:

Option	Beschreibung
Launch IT-Grundschutz (10. EL)	Mit dieser Option starten Sie die zehnte Ergänzungslieferung der BSI-Grundschutzkataloge.
Launch IT-Grundschutz (11. EL)	Startet entsprechend die aktuellste Ergänzungslieferung der IT-Schutzkataloge.
Verbose IT-Grundschutz results	Schaltet die „geschwätzige“ Ausgabe für die IT-Grundschutzkataloge ein, wobei eine sehr detaillierte Ausgabe erzeugt wird.



Wenn Sie die IT-Grundschutzprüfungen aktivieren, finden Sie im Bericht die relevanten Informationen.

4.2.5 CPE-base Policy Check

CPE steht für Common Product Enumeration und ist ein strukturiertes Namensschema für informationstechnische Systeme, Plattformen und Pakete. CPE ist also nicht anderes als eine eindeutige Kennung für nahezu jede Software, für die eine Schwachstelle bekannt ist.

Das CPE-Verzeichnis wird von MITRE (*http://www.mitre.org*) und NIST (National Institute of Standards and Technology, *http://www.nist.gov*) betreut. Die Spezifikationen finden Sie unter *http://cpe.mitre.org/specification/index.html.*

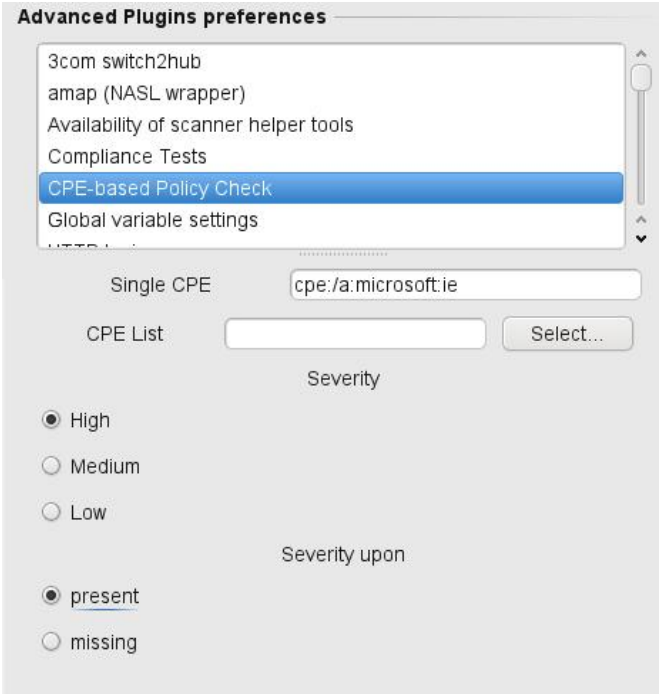
Eine CPE besteht neben der Quellenangabe aus drei Teilen:

- **Allgemeines**: In diesem Teil werden die Hardware, das Betriebssystem und die Anwendung aufgeführt.

- **Hersteller**: Hier wird der Entwickler angegeben.
- **Weitere Informationen**: Hier finden Sie das Produkt, die Version, Update- und Editionsinformationen, Sprachvariante

Ein Beispiel:

```
cpe:/a:mysql:community_server:5.0.41
```



Die Einstellungen für die CPE-basierten Tests.

Mit diesen Einstellungen des Bereichs *CPE-based Policy Check* bestimmen Sie, ob, und wenn ja, welche CPE-basierten Tests durchgeführt werden.

Wenn Sie CPEs in Ihren Scans verwenden, ist es einfach herauszufinden, ob ein bestimmtes Programm Probleme bereitet oder nicht. Die Einstellungen im Überblick:

Option	Beschreibung
Single CPE	Geben Sie hier einen Common Product Enumeration-Eintrag an, wenn Sie das System auf eine spezifische Software-Schwachstelle hin überprüfen wollen.
CPE List	Wenn Sie eine ganze Liste von CPE-Einträgen überprüfen wollen, geben Sie hier die Liste mit den Einträgen an. Führen Sie die Einträge in einer kommaseparierten Liste auf.
Severity	Hier bestimmen Sie, welchen Schweregrad die Verwundbarkeit besitzt. Sie haben die Wahl zwischen den drei Optionen *Low*, *Medium* und *High*.
Severity upon	Hier haben Sie die Wahl zwischen den beiden Optionen *present* und *missing*.

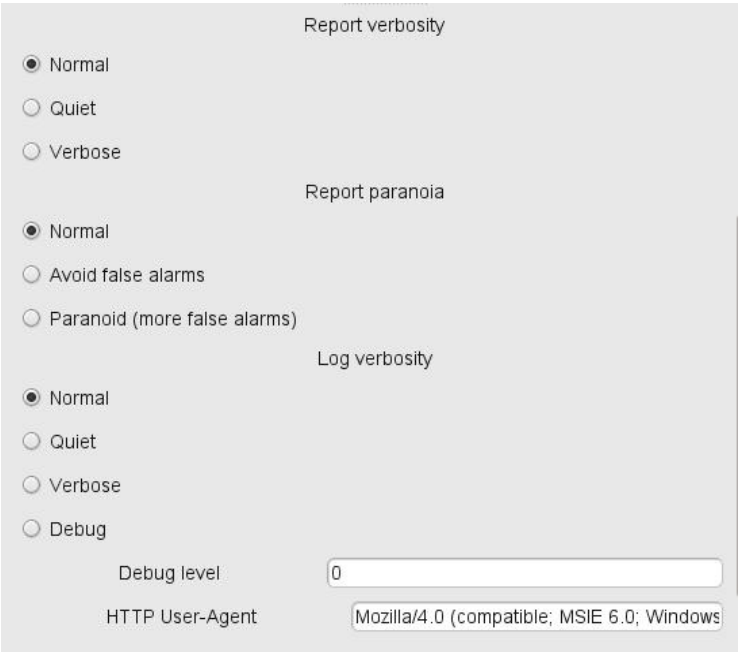
4.2.6 Global variable settings

Als Nächstes finden Sie in dem Auswahlmenü die Konfigurationen des Bereichs *Global variable settings*. Damit werden die Einstellungen insbesondere der Bericht- und Logfileausgabe vorgenommen.

Option	Beschreibung
Enable CGI scanning	Wenn Sie diese Option aktivieren – sie ist standardmäßig aktiv – versucht sich OpenVAS auch am Testen von CGI-Skripts, die vorzugsweise auf Webservern ausgeführt werden. Beachten Sie, dass derlei Tests viel Zeit in Anspruch nehmen. Wenn Sie also nicht zwingend auf diese Tests angewiesen sind, sollten Sie diese Option deaktivieren. Das führt zu einer deutlich schnelleren Ausführung.
Network type	Dieses Auswahlmenü stellt Ihnen drei Netzwerktypen zur Auswahl, die Sie Ihren Tests unterziehen wollen. Sie haben die Wahl zwischen folgenden Optionen: • Mixed (use RFC 1918) • Private LAN • Public WAN (Internet) • Enable experimental scripts • Thorough tests (slow)

	Die Optionen sind selbsterklärend. Mit der Standardeinstellung *Mixed* können Sie nichts falsch machen. Verwenden Sie diese insbesondere dann, wenn Sie ein lokales Netzwerk mit mehreren Routern scannen wollen. Wenn Sie die Option *Enable experimental scripts* aktivieren, werden auch Testskripts durchgeführt, die als experimentell gekennzeichnet sind. Das Aktivieren der Option *Thorough tests (slow)* bewirkt, dass verschiedene Skripts neben den Standardtests weitere Untersuchungen anstellen. Das kann zu einer spürbar längeren Testdauer führen.
Report verbosity	Hier legen Sie die „Geschwätzigkeit" der Berichte fest. Es stehen folgende Optionen zur Verfügung: • *Normal* • *Quit* • *Verbose* *Quit* und *Verbose* liefern weniger bzw. sehr detaillierte Informationen, als es bei der Standardkonfiguration *Normal* der Fall ist.
Report paranoia	Mit diesem Schalter können Sie die Empfindlichkeit der Berichtfunktionen anpassen. Es stehen folgende Optionen zur Auswahl: • *Normal* • *Avoid false alarms* • *Paranoid* Es empfiehlt sich, die Standardeinstellung *Normal* beizubehalten.
Log verbosity	Mit dieser Option legt man den Umfang der Protokolle fest. Es stehen folgende Optionen zur Verfügung: • *Normal* • *Quiet* • *Verbose*

	• *Debug* Die Informationsmenge steigt von oben nach unten.
Debug level	Mit dieser Einstellung legen Sie den Debug-Level der Skripts fest. Mit dem Standardwert 0 ist die Funktion deaktiviert. Ein Hochsetzen ist nur bei der Fehlersuche sinnvoll.
HTTP User Agent	Hier geben Sie den User-Agent an.

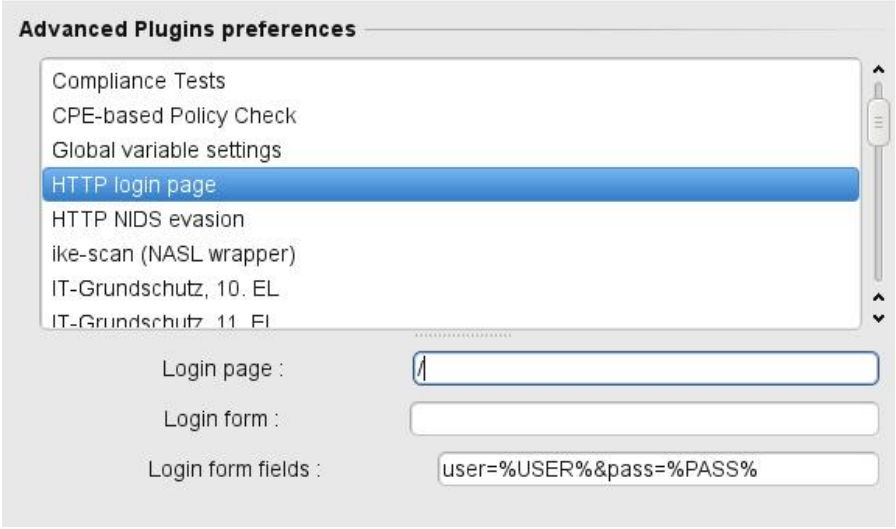


Die Konfiguration verschiedener globaler Einstellungen.

4.2.7 HTTP login page

Als Nächstes können Sie die für das Einloggen über ein typisches Web-Formular notwendigen Daten hinterlegen. Dazu stehen Ihnen drei Einstellungen zur Verfügung:

Option	Beschreibung
Login page	Hier geben Sie den HTTP-Pfad für das Formular ein, das man für das Einloggen verwenden will. OpenVAS versucht zunächst, sich einzuloggen, und führt dann seine Tests durch.
Login form	In diesem Eingabefeld geben Sie das Formular gezielt an, das OpenVAS ansteuern soll.
Login form fields	Ist für den Zugriff auf einen Webserver die Authentifizierung erforderlich, so kann man in diesem Feld die dafür benötigten Informationen hinterlegen. Dabei greift OpenVAS auf die Angaben im Abschnitt *Login configurations* zurück und ersetzt die Variablen *%USER%* und *%PASS%* durch die dort hinterlegten Daten.

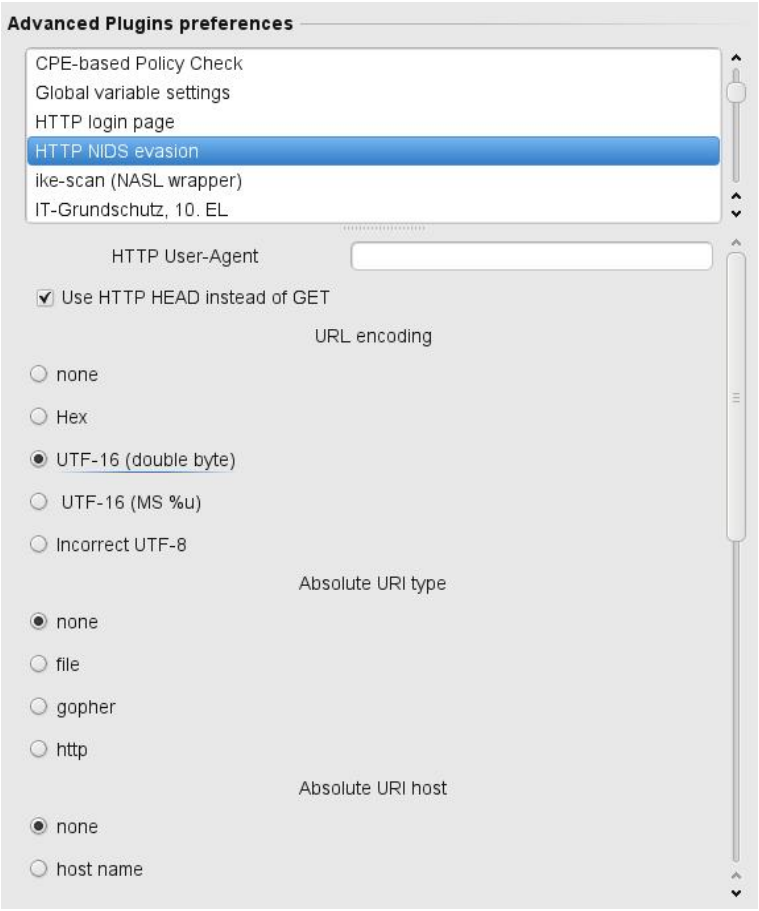


Die Konfiguration des HTTP-Log-ins.

4.2.8 HTTP NIDS evasion

In geschützten Umgebungen kommen oftmals auch Intrusion-Detection-Systeme zum Einsatz. Erkennen diese Ihre Scan-Aktivitäten, so werden deren Logdateien mit unnützen Informationen zugemüllt, die man, wenn man die Scans selbst durchführt, wahrlich nicht gebrauchen kann.

Mit den Einstellungen des Abschnitts *HTTP NIDS evasion* versucht OpenVAS, die Erkennungsmechanismen zu umgehen. Allerdings sollte man auch wissen, dass sich dabei Verfälschungen der Berichte ergeben können.



Die NIDS-Einstellungen.

Die Einstellungen im Einzelnen:

Option	Beschreibung
HTTP User Agent	In diesem Eingabefeld legt man User-Agent-spezifische Angaben fest, die OpenVAS verwenden soll.
Use http HEAD instead of GET	Aktiviert man diese Option, so verwendet OpenVAS die HTTP-HEAD-Methode für Requests statt der typischen GET-Methode.
URL Encoding	URLs können mit verschiedenen Zeichensätzen codiert werden. Gültige Werte sind: • *none* • *Hex* • *UTF-16 (double byte)* • *UTF-16 (MS %u)* • *Incorrect UTF-8* Die Vorgabe *none* kann in der Regel beibehalten werden.
Absolute URI type	Hier legt man den URI-Typ fest, den OpenVAS verwendet. Mögliche Werte sind: • *none* • *file* • *gopher* • *http* Der Standard ist *none*.
Absolute URI host	Hier wählt man den URI-Host, den OpenVAS an das Ziel übermittelt. Mögliche Alternativen sind: • *none* • *host name* • *host IP* • *random name*

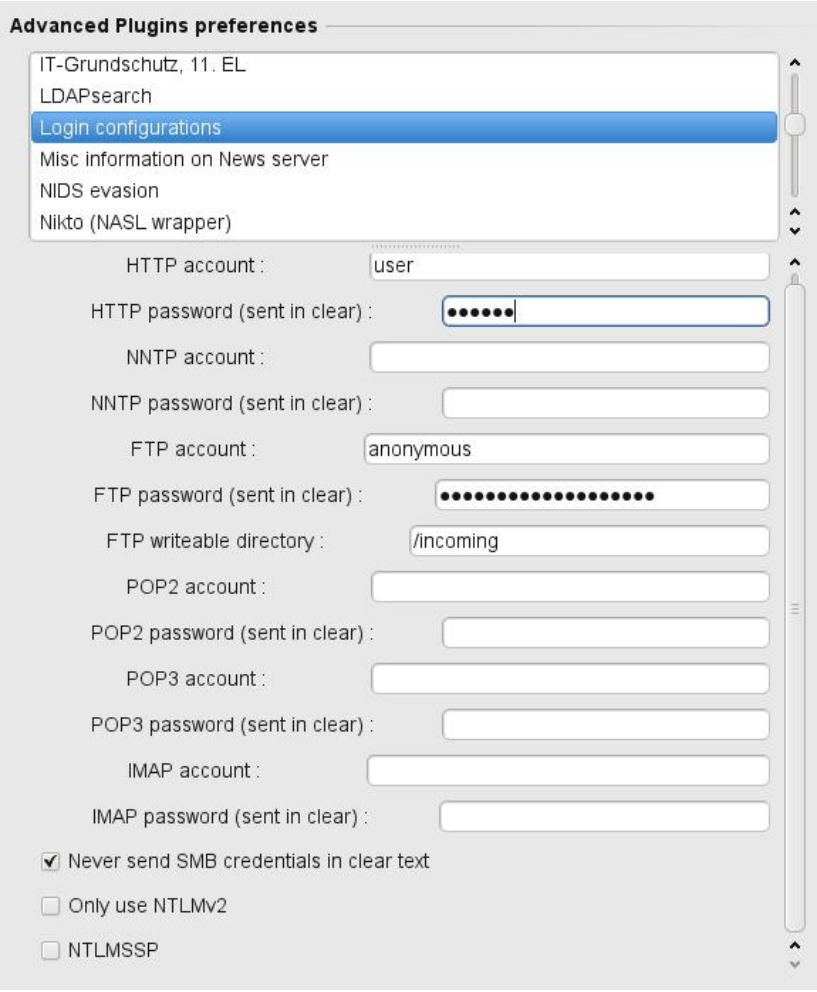
	• *random IP* Standardmäßig wird keine Information übertragen.
Double slashes	Wenn Sie dieses Kontrollkästchen aktivieren, so werden alle einfachen Schrägstriche (/) in doppelte Schrägstriche verwandelt (//).
Reverse Traversal	Mit der Reverse Traversal versucht man, das IDS durch eine erweiterte URL zu irritieren, wobei statt */cgi-bin/test-cgi* die URL */cgi-bin/redherring/../test.cgi* verwendet wird. Hier kann man diese Funktion mit *none* ausschalten und die Standardvariante (*basic*) verwenden. Außerdem kann man eine lange URL nutzen und diese mit weiteren Einstellungen versehen.
Self-reference	Diese Option unterstützt selbstreferenzierende Tests. Dabei wird aus jedem / ein /./.
Premature request ending	Diese Option ist mit der Funktion *Reverse traversal* vergleichbar, allerdings enthält der Verzeihnisname Zeichen, die nach CR LF übersetzt werden können.
CGI.pm semicolon separator	Diese Option verwendet ' ' anstelle des Und-Zeichens in der Suchabfrage.
Parameter hiding	Dies ist eine weitere Form der Option *Reverse traversal*. Der Teil eines Verzeichnisses, der die Zeichenfolge %3F enthält, kann beispielsweise in ein Fragezeichen (?) übersetzt werden.
Dos/Windows syntax	Wenn Sie dieses Kontrollkästchen aktivieren, so verwendet OpenVAS den Backslash (\) statt des normalen Schrägstrichs (/).
Null method	Aktivieren dieser Option bewirkt, dass OpenVAS die Zeichen %00 zwischen die Methode und die URI einfügt.
TAB separator	Wenn Sie diese Option aktivieren, wird ein TAB anstelle eines Leerzeichens zwischen der Methode, der URL und HTTP verwendet.
HTTP/0.9 requests	Diese Option sorgt dafür, dass OpenVAS HTTP-0.9-kompatible Requests nutzt. Dabei werden lediglich die Methode und die URI angegeben, das HTTP-Versionsfeld jedoch nicht.

Force protocol string	In diesem Eingabefeld können Sie OpenVAS vorgeben, welchen Protokoll-String der Scanner bei seinen Requests verwenden muss.
Random case sensitivity (Nikto only)	Díese Option ist wichtig, wenn Sie den CGI-Scanner Nikto (*http://cirt.net*) in OpenVAS integrieren und der per Zufallsprinzip Groß- und Kleinschreibung verwendet.

4.2.9 Login configurations

Auf dem Dialog *Login configurations* hinterlegen Sie die Benutzernamen und Passwörter für das Testen von gängigen Diensten wie HTTP, FTP etc. Die Einstellungen im Überblick:

Option	Beschreibung
HTTP account	Hier geben Sie den Benutzernamen für den HTTP-Dienst an.
HTTP password	Hier hinterlegen Sie das zugehörige Passwort.
FTP account, password etc.	Entsprechend stehen Eingabefelder für die Dienste NNTP, POP2, POP3 und IMAP zur Verfügung. Beachten Sie, dass die Passwörter als Klartext übermittelt werden.
Never send SMB credentials in clear text	Diese Option ist standardmäßig aktiviert und sorgt für die verschlüsselte Übermittlung der Zugangsdaten.
Only use NTLMv2	NTLM ist ein Authentifizierungsschema, das von Microsoft stammt und überwiegend in deren Produkten zum Einsatz kommt. Es ist in zwei Varianten verfügbar: NTLM und NTLM2. Version 2 ist sicherer. Hier können Sie die Kommunikation auf NTLM2 beschränken. Das macht nur Sinn, wenn Sie zuverlässig wissen, dass auch die zu scannenden Systeme alle diese Varianten unterstützen.
NTLMSSP	Erlaubt die Verwendung von NTLMSSP (NT LAN Manager Security Support Provider). Das ist ein Binärprotokol von Microsoft, das NTLM-Authentifizierungen durchführt.

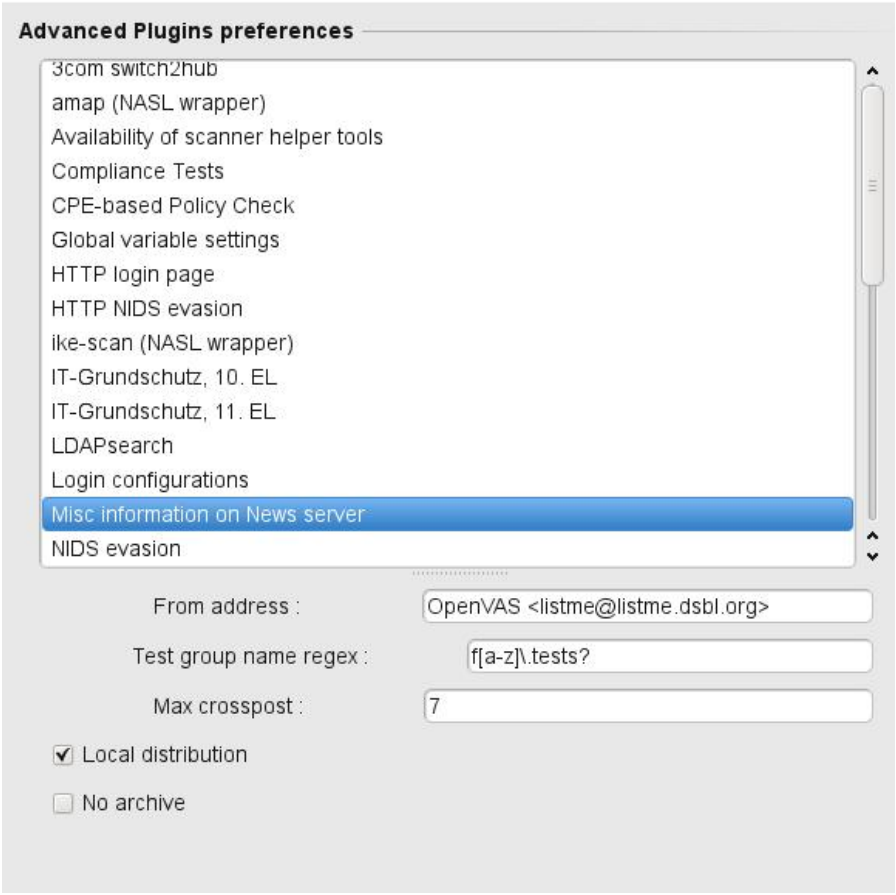


Die Konfiguration der verschiedenen Log-in-Optionen, mit denen sich OpenVAS Zutritt zu verschiedenen Diensten verschafft.

4.2.10 Misc information an News server

Es folgen die Einstellungen für das Testen von Newsservern. Die verfügbaren Optionen fasst nachstehende Tabelle zusammen:

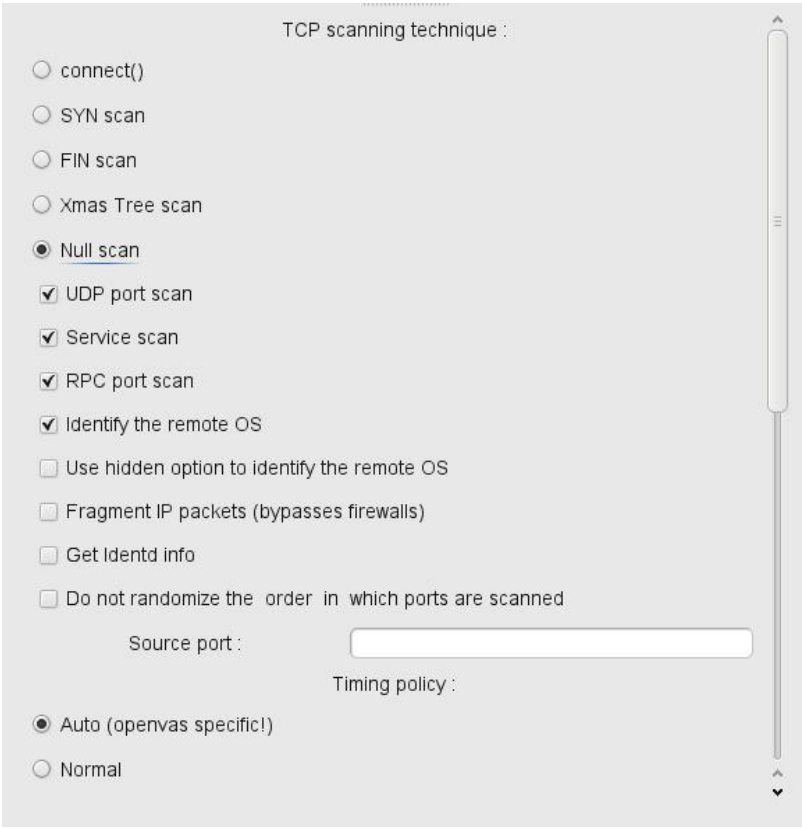
Option	Beschreibung
From address	Hier können der Tester und seine E-Mail-Adresse hinterlegt werden.
Test group name regex	In diesem Eingabefeld können Sie den regulären Ausdruck für das Testen bestimmter Gruppen verwenden, deren Newsgroup-Bezeichnung dem Ausdruck entspricht.
Max crosspost	OpenVAS versendet beim Testen von Newsservern Testnachrichten. Hier legen Sie den maximalen Wert für Crosspostings fest.
Local distribution	Diese Option begrenzt das Versenden von Testnachrichten für das lokale Verteilen.
No archive	Wenn Sie diese Option aktivieren, so versucht OpenVAS zu verhindern, dass die Testnachrichten archiviert werden.



Die Konfiguration des Newsserver-Tests.

4.2.11 Nikto

Wenn Sie auch den Nikto-Scanner verwenden, können Sie eine Anpassung vornehmen: *Force Scan even without 404s*. Wenn Sie dieses Kontrollkästchen aktivieren, werden Scans erzwungen.



OpenVAS stellt Ihnen eine Fülle an Einstellungen für die Steuerung von Nmap zur Verfügung.

4.2.12 Nmap

In OpenVAS ist auch Nmap (Network Mapper), der Klassiker unter den Open-Source-Werkzeugen für die Netzwerkanalyse und Sicherheitsüberprüfung, integriert. Dieses Tool wurde entwickelt, um große Netzwerke schnell zu scannen, funktioniert aber auch bei einzelnen Hosts sehr gut.

Nmap verwendet rohe IP-Pakete, um festzustellen, welche Hosts im Netzwerk verfügbar sind, welche Dienste (Anwendungsname und -version) diese Hosts bieten, welche Betriebssysteme (und Versionen davon) darauf laufen, welche Art von Paketfiltern/-Firewalls verwendet werden. Nmap kann Dutzende weitere Eigenschaften auslesen.

Abhängig von den verwendeten Optionen gibt Nmap eine Liste der gescannten Ziele mit zusätzlicher Information zu jedem aus. Für weitere Details sei auf die Website des Nmap-Projekts (*http://nmap.org*) verwiesen.

Option	Beschreibung
connect()	Der TCP-Connect-Scan ist der standardmäßig eingestellte TCP-Scan-Typ, sollten SYN-Scans nicht möglich sein. Das ist dann der Fall, wenn der Benutzer kein Recht hat, rohe Pakete zu senden, oder wenn er IPv6-Netzwerke scannt.
SYN scan	Der SYN-Scan ist die Standardeinstellung und die beliebteste Scan-Methode. Diese Scan-Variante wird schnell durchgeführt und scannt dabei Tausende von Ports pro Sekunde, wenn das Netzwerk schnell ist und nicht von einer intrusiven Firewall behindert wird. Ein weiterer Vorteil: Der SYN-Scan ist relativ unauffällig, da er TCP-Verbindungen niemals abschließt.
FIN scan	Die FIN-, Xmas- und Null-Scanverfahren nutzen ein subtiles Schlupfloch im TCP-RFC aus, um zwischen offenen und geschlossenen Ports zu unterscheiden. Diese Varianten verwenden nur das TCP-FIN-Bit.
Xmas Tree scan	Diese Option setzt die FIN-, PSH- und URG-Flags und beleuchtet das Paket wie einen „Weihnachtsbaum“.
Null scan	Bei dieser Option werden keinerlei Bits gesetzt.
UDP-Scan	Wenngleich die meisten bekannten Internetdienste über das TCP-Protokoll laufen, sind UDP-Dienste weitverbreitet. Drei der häufigsten sind DNS, SNMP und DHCP. Mit dieser Option führen Sie einen UDP-Scan durch. Die sind im Allgemeinen langsamer und schwie-

	riger als TCP-Scans, daher werden werden diese Ports von manchen Sicherheitsprüfern einfach ignoriert. Das ist problematisch, denn angreifbare UDP-Dienste sind recht häufig und können mit dieser Option einfach aufgedeckt werden.
Service scan	Hat Nmap mit einer Scan-Methoden TCP- und/oder UDP-Ports entdeckt, kann es zusätzlich auch eine Service- und Versionskennung dieser Ports durchführen. Das Ziel: Mehr darüber zu erfahren, was tatsächlich auf diesen Ports läuft. Die Datenbank in nmap-service-probes enthält Testpakete für die Abfrage verschiedenster Dienste und Ausdrücke für den Vergleich und das Parsen der Antworten. Nmap versucht, das Dienstprotokoll zu bestimmen (z. B. FTP, SSH, Telnet, HTTP), aber auch Anwendungsnamen, Versionsnummer, Hostnamen, den Gerätetyp, das Betriebssystem und wenn möglich, verschiedene weitere Details.
RPC scan	Diese Option könenn Sie ebenfalls mit verschiedenen Port-Scan-Methoden von Nmap kombinieren. Sie nimmt alle offenen TCP-/UDP-Ports und überflutet diese mit NULL-Befehlen um festzustellen, ob es RPC-Ports sind, und wenn ja, welches Programm und welche Versionsnummer darauf läuft.
Identify the remote OS	Eine der Stärken von Nmap ist die Erkennung von Betriebssystemen mithilfe von TCP/IP-Stack-Fingerprinting. Dabei sendet das Tool eine Reihe von TCP- und UDP-Paketen an das Zielsystem und untersucht jedes Antwort-Bit. Ist der Test durchlaufen, vergleicht Nmap die Ergebnisse mit seiner Datenbank in *nmap-os-db* mit über 1.000 bekannten Betriebssystem-Fingerprints und gibt auf deren Grundlage die Details des identifzierten Betriebssystems aus.
Use hidden option to identify the remote OS	Nmap stellt Ihnen verschiedene versteckte Optionen und Funktionen für die Identifikation des Betriebssystems aufseiten des Zielsystems zur Verfügung. Diese aktivieren Sie hier.
Fragment IP packets	Nutzt die Fragmentierungsmöglichkeiten.
Get Identd info	Durch das Aktivieren dieser Option wird die identd-Information laut RFC 1413 über die laufenden Prozesse auf dem Ziel-System eingeholt.

Do not randomize the order in which ports are scanned	Nmap fragt die Ports standardmäßig nach dem Zufallsprinzip ab. Wenn Sie diese Option aktivieren, erfolgt die Scan-Reihenfolge sequenziell. Nmap scannt standardmäßig für jedes Protokoll die 1000 meistbenutzten Ports.
Source Port	Um Firewalls zu umgehen bzw. Intrusion-Detection-Systeme zu täuschen, können Sie hier den Quell-Port bestimmen.
Timing policy	Die Nmap-Entwickler legen viel Wert auf eine gute Performance des Tools. Ein Standardscan (*nmap <hostname>*) eines Hosts in einem lokalen Netzwerk dauert laut Angaben der Entwickler eine Fünftelsekunde. Verschiedene Tests können Sie mit Timing-Parametern steuern, um eine möglichst schnelle Abarbeitung zu erreichen. Mittels eines Templates lassen sich diese steuern. Nmap stellt Ihnen verschiedene Templates zur Verfügung: paranoid (0), sneaky (1), polite (2), normal (3), aggressive (4) und insane (5). Die ersten beiden sind für die Umgehung von IDS gedacht. Der Polite-Modus verlangsamt den Scan, um wenig Bandbreite und Ressourcen auf dem Zielrechner zu beanspruchen. Der Normal-Modus ist der Standardwert. Im Aggressive-Modus werden die Tests beschleunigt.

Den Abschluss der Nmap-Konfiguration bilden verschiedene benutzerdefinierbare Einstellungen:

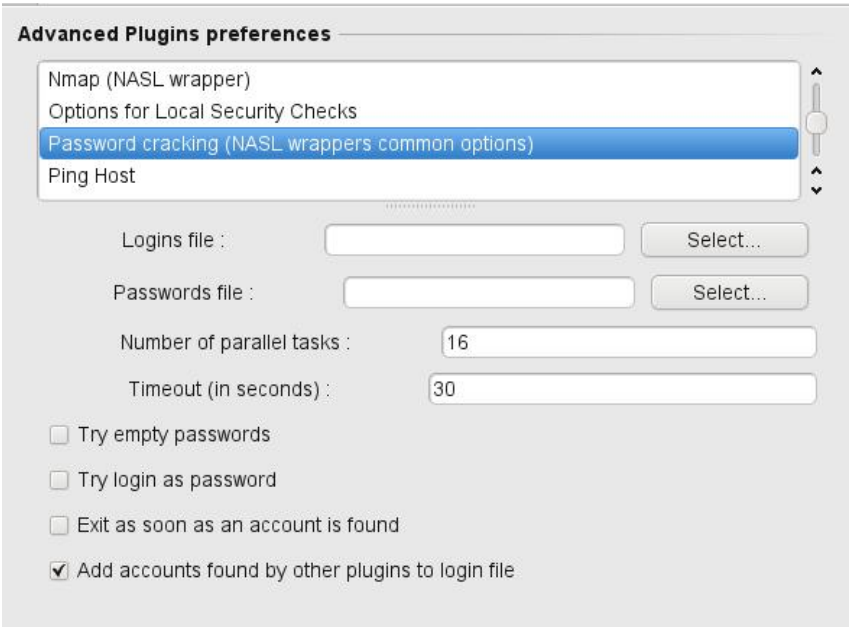
Option	Beschreibung
Host Timeout	Bei manchen Hosts braucht es sehr lange, sie zu scannen. Mit dieser Einstellungen geben den Sie die maximale Wartezeit an.
Min RTT Timeout	Mit dieser und den zwei folgenden Einstellungen bestimmen Sie die Timeouts von Testpaketen. Die Anpassung dieses Werts kann sinnvoll sein, wenn ein Netzwerk so unzuverlässig ist, dass selbst die Standardeinstellungen von Nmap zu aggressiv sind.
Max RTT Timeout	Bestimmt den maximalen Timeout-Wert. 100 Millisekunden gelten als ein vernünftig aggressiver Wert.

Initial RTT Timeout	Bestimmt den Anfangs-Timeout-Wert.
Ports scanned in parallel (max)	Mit dieser und der nächsten Einstellung steuern Sie die Gesamtanzahl an Testpaketen, die für eine Host-Gruppe anstehen dürfen. Sie werden beim Port-Scanning und bei der Host-Entdeckung benutzt. Abhängig von der Netzwerk-Performance berechnet Nmap standardmäßig eine immer wechselnde ideale Parallelität. Diese wird gerne auf den Wert 1 gesetzt, um zu verhindern, dass Nmap mehr als ein Testpaket auf einmal an Hosts sendet.
Ports scanned in parallel (min)	Meistens setzt man diesen Wert auf eine Zahl größer als eins, um Scans von langsamen Hosts oder Netzwerken zu beschleunigen.
Minimum wait between probes	Mit dieser Einstellung geben Sie die Mindestunterbrechung zwischen zwei Tests an.
File containing grepable results	Nmap kann seine Ausgabe in fünf verschiedenen Formaten erzeugen. Das Standardformat heißt interaktive Ausgabe und wird an die Standardausgabe (stdout) gesendet. Ein wichtiges Format wird auch als grepbare Ausgabe bezeichnet. Hier bestimmen Sie die Dateien, in der die Daten landen.
Do not scan targets not in the file	Durch Aktivieren dieser Optionen werden die Ziele, die nicht in der Zieldatei aufgeführt werden, auch nicht gescannt.
Run dangerous port scans even if safe checks are set	Wenn Sie diese Option aktivieren, werden potenziell gefährliche Plug-ins auch dann ausgeführt, wenn Sie in den Scan-Optionen *Safe Checks* aktiviert haben. Sie hebeln damit diese Einstellung aus.

4.2.13 Options for Local Security Checks

Für die Durchführung von lokalen Sicherheitstests stehen Ihnen lediglich zwei Anpassungsmöglichkeiten zur Verfügung:

Option	Beschreibung
Also use find command to search for applications	Wenn Sie diese Option aktivieren, wird auch das Kommando *find* für die Suche nach bestimmten Anwendungen verwendet. Es kann helfen, gesuchte Applikationen einfacher aufzuspüren.
Descend directories on other filesystems	Die Verzeichnisstruktur anderer Dateisysteme kann durch Aktivieren dieses Kontrollkästchens absteigend abgearbeitet werden.



OpenVAS besitzt auch einen Passwort-Cracker, den Sie in den erweiterten Plug-in-Einstellungen anpassen können.

4.2.14 Password cracking

OpenVAS kommt auch mit einem Passwort-Cracker, mit dem Sie versuchen können, sich auf Grundlage zweier anzugebender Dateien (eine für die Log-ins, die andere für die Passwörter) sich Zugang zu Systemen zu verschaffen.

Die Einstellungen für diesen Bereich:

Option	Beschreibung
Logins file	Geben Sie hier die Datei an, in der die Log-ins hinterlegt sind.
Passwords file	Hier geben Sie entsprechend die Datei an, in der die Passwörter zu finden sind.
Number of parallel tasks	Geben Sie hier die Anzahl an parallel ausgeführten Crack-Versuchen an.
Timeout	Der Timeout-Wert ist mit 30 Sekunden standardmäßig hoch gesetzt.
Try empty passwords	Wenn Sie dieses Kontrollkästchen aktivieren, versucht der Cracker es auch mit leeren Passwörtern.
Try login as password	Der Cracker kann auch für die Log-ins und die zugehörigen Passwörter identische Zeichenketten verwenden.
Exit as soon as an account is found	Wenn Sie dieses Kontrollkästchen aktivieren, beendet der Cracker seine Ausführung, sobald ein Account gefunden ist, mit dem Sie sich Zugang zu dem Zielsystem verschaffen können.
Add accounts found by other plugins to login file	Finden andere Testskripts Accounts, so werden diese standardmäßig auch in der Log-in-Datei des Passwort-Crackers eingetragen.

4.2.15 Ping the remote host

Die Ping-Utility ist eines der einfachsten und wichtigsten Hilfsmittel beim Scannen von entfernten Systemen. Es dient in erster Linie dazu, das Vorhandensein von Hosts zu verifizieren.

Auf der Seite *Ping the remote host* stehen Ihnen folgende Einstellungen zur Verfügung:

Option	Beschreibung
Report about unreachable Hosts	Wenn Sie diese Option aktivieren, berichtet Ping über nicht erreichbare Systeme.
Mark unreachable Hosts as dead	Nicht erreichbare Systeme werden als „tot“ gekennzeichnet und daher wird auch kein Scan-Versuch unternommen.

4.2.16 Pnscan

Pnscan ist ein multi-threaded Portscanner, der ein großes Netzwerk sehr schnell scannen kann. Er bringt längst nicht alle Fähigkeiten mit, die Nmap zu bieten hat, ist dafür aber deutlich schneller.

Für die Konfiguration dieses Tools stehen Ihnen zwei Optionen zur Verfügung:

Option	Beschreibung
Pnscan Timeout	In diesem Eingabefeld geben Sie den Timeout-Wert an.
Pnscan concurrent worker Threads	Mit dieser Eingabe bestimmen Sie, wie viele Pnscan-Threads gleichzeitig ausgeführt werden.

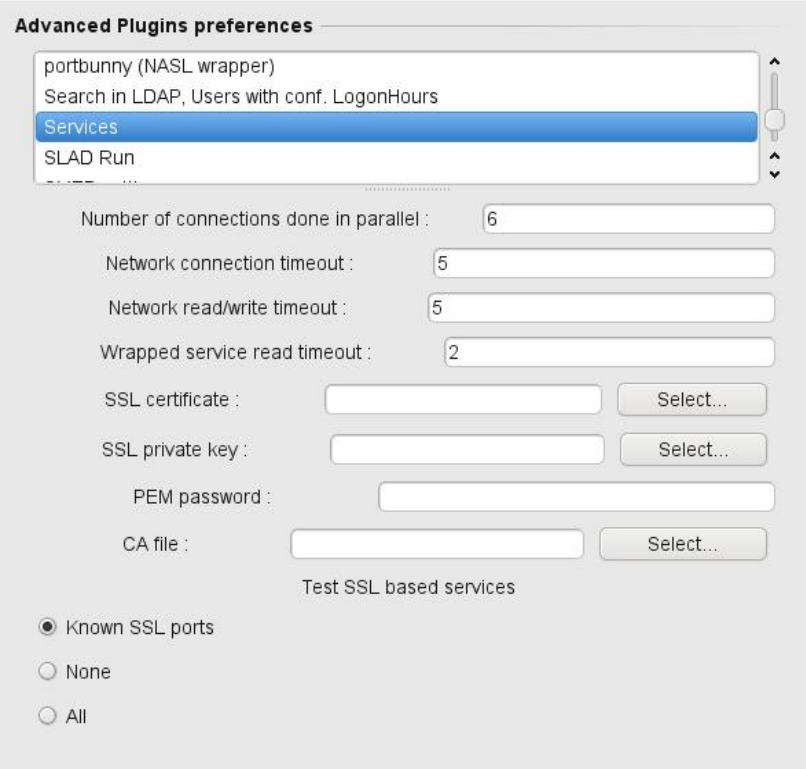
4.2.17 portbunny

Mit portbunny (*http://portbunny.recurity.com*) besitzt OpenVAS einen weiteren Port-Scanner, der auf eine hohe Geschwindigkeit getrimmt ist. Im Unterschied zu Nmap konzentriert sich dieses Tool auf seine Kernfunktion: Das Scannen von Ports.

Für die Anpassung dieses Tools steht Ihnen lediglich eine Konfigurationsmöglichkeit zur Verfügung: Mit *Wait longer for triggers to return* wartet der Scanner länger auf Reaktionen der Ziele.

4.2.18 Search in LDAP

Für dieses Plug-in können Sie zwei Anpassungen vornehmen: Sie können den Namen und die Einheit des Testusers angeben.



Die Service-Konfiguration.

4.2.19 Services

Der nächste Bereich trägt die Bezeichnung *Services*. Auf der zugehörigen Seite können Sie verschiedene dienstübergreifende Einstellungen bearbeiten. Konkret sind es die folgenden:

Option	Beschreibung
Number of connections done in parallel	Hier legen Sie fest, wie viele Verbindungen pro Dienst OpenVAS gleichzeitig zu einem Ziel aufbaut. Der Standardwert ist 6.
Wrapped service read timeout	Hier legen Sie die Wartezeit in Sekunden fest, die OpenVAS auf die Reaktion eines übernommenen Dienstes wartet.

Network connection timeout	In diesem Eingabefeld geben Sie die Dauer in Sekunden an, die OpenVAS auf eine Netzwerkverbindung zu einem Ziel wartet.
Network read/write timeout	Hier legen Sie den Timeout-Wert für das Lesen bzw. Schreiben fest. Der Standardwert ist wie oben 5 Sekunden.
SSL certificate	Hier legen Sie den Pfad zum SSL-Zertifikat fest.
SSL private key	Hier geben Sie den Pfad zum privaten SSL-Schlüssel an.
PEM password	In diesem Textfeld hinterlegen Sie das PEM-Passwort.
CA file	Hier geben Sie den Pfad zur CA-Datei an.
Test SSL based services	Hier legen Sie fest, welche SSL-basierten Dienste getestet werden. Über das Auswahlmenü haben Sie die Wahl zwischen drei Einstellungen: • *Known SSL ports* • *None* • *All*

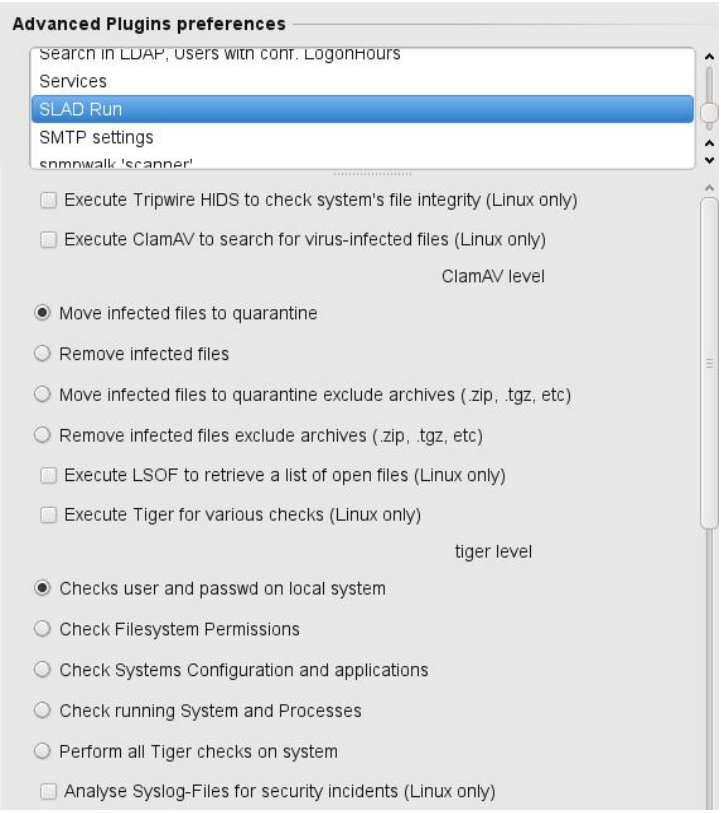
4.2.20 SLAD

Mit Security Local Auditing Daemon (SLAD) ist ein Dienst in OpenVAS integriert, der andere Programme einbindet und eine einheitliche Schnittstelle zu ihren Ausgaben zur Verfügung stellt.

In der aktuellen Version enthält SLAD die folgenden Plug-ins:

- **Chkrootkit**: Dieses Programm untersucht das lokale System auf Anzeichen für installierte Rootkits.
- **ClamAV**: Freier Virenscanner für Linux. Sie können dabei steuern, ob Archive gescannt werden oder nicht und ob infizierte Dateien entfernt werden sollen oder in einem Quarantäne-Bereich isoliert werden sollen.
- **John**: „John the ripper“ ist ein schneller Passwort-Cracker. Sein Ziel ist es, schwache Benutzerpasswörter zu finden, die die Sicherheit des Systems beeinträchtigen könnten.
- **Lsof**: Dieses Unix-Systemprogramm zeigt eine Liste der zurzeit auf dem System offenen Dateien an und welche Programme diese Dateien nutzen. Es hilft, ungewollte Aktivitäten auf dem System zu identifizieren.

- **Tiger**: Eine Sammlung zur Analyse der Sicherheit des Rechners. Dabei wird eine Vielzahl der Sicherheitstests durchgeführt.
- **Tripwire**: Dieses Programm prüft die Integrität von Dateien. Beim ersten Aufruf erstellt Tripwire eine Datenbank mit Hashwerten der Systemdateien, die es bei zukünftigem Aufruf zum Vergleich heranzieht. Veränderungen am System lassen sich auf diese Weise leicht feststellen.
- **Snort**: Snort dient der Analyse und Überwachung des Datenverkehrs in IP-Netzwerken in Echtzeit. Damit ist es in der Lage, eine Vielzahl von Angriffen wie etwa Pufferüberläufe, verdeckte Portscans, CGI-Angriffe, SMB-Scans oder versuchte Betriebssystemerkennungen, zu entdecken. Snort kann in geringem Umfang auch Gegenmaßnahmen ergreifen.



Ein Blick auf einen Teil der SLAD-Einstellungen.

- **LMSensors**: LMSensors erlaubt das Auslesen von Ereignissen der Hardware-Überwachung.
- **LogWatch**: Dieses Tool liest Ereignisse aus den Systemprotokollen aus, etwa aus den unter */var/log/* vorhandenen Dateien. Dabei werden alle wichtigen Informationen, wie Benutzeranmeldungen, SSH- und PAM-Sitzungen etc., gefiltert, zusammengefasst und an den SLAD-Dienst weitergegeben.
- **TrapWatch**: Bei diesem Werkzeug handelt es sich um eine spezielle Version von LogWatch, die SNMP-Meldungen („Traps“) von Geräten protokolliert.

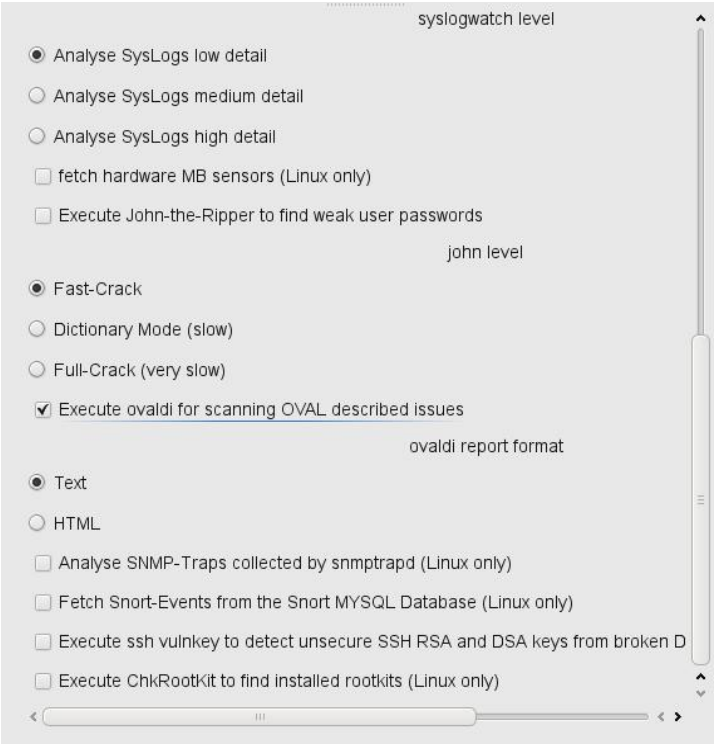
Für weitere Details sei auf den SLAD-Admin-Guide (*http://www.dnsystems.org/projects/slad/slad-dev-and-admin-guide.pdf*) verwiesen.

Aufgrund der umfangreichen Funktionalität hat SLAD auch vergleichsweise viele Konfigurationsmöglichkeiten zu bieten, insbesondere für das Zusammenspiel mit oben genannten Anwendungen:

Option	Beschreibung
Allgemeine Einstellungen	
Execute Tripwire HIDS to check system‘s file integrity	Über SLAD ist die Ausführung von Tripwire möglich, um die Integrität der zu scannenden Systeme zu prüfen. Das funktioniert allerdings nur mit Linux-Systemen.
Execute ClamAV to search for virus-infected files	SLAD kann den Virenscanner ClamAV ausführen, um die zu untersuchenden Systeme auf mögliche Dateiinfektionen zu prüfen.
ClamAV-Einstellungen	
Move infected files to quarantine	Mit dieser Standardeinstellung werden infizierte Dateien in die Quarantäne verschoben, damit sie dort kein Unheil mehr anstellen können.
Remove infected files	Mit dieser Option entfernen Sie infizierte Dateien. Setzen Sie diese Option mit Bedacht ein.
Move infected files exclude archives	Verschiebt infizierte Dateien in den Quarantänebereich, schließt Archive allerdings aus.
Execute LSOF to retrieve a list of open files	Wenn Sie diese Option aktivieren, wird das Tool LSOF ausgeführt, um die Liste der offenen Dateien einzulesen.
Execute Tiger for	Zusätzlich zu den Viren-Checks können Sie Tiger star-

various checks	ten, um weitere Sicherheitstest durchzuführen.
Tiger-Einstellungen	
Checks user and password on local system	Dieser Test deckt Benutzerkonten ab. Dabei werden verschiedene Dinge geprüft, beispielsweise die E-Mail-Weiterleitung, FTP-Zugänge und dergleichen mehr.
Check Filesystem Permission	Mit dieser Option werden die Zugriffsrechte von Benutzern und Gruppen auf bestimmte wichtige Dateien und Verzeichnisse überprüft.
Check System Configuration and applications	Wenn Sie diese Option wählen, sucht Tiger nach Schwachstellen und Fehlern in verbreiteten system- und anwendungsspezifischen Konfigurationsdateien.
Check running System and Process	Diese Testvariante sucht nach offenen gelöschten Dateien, Prozessen, die auf ankommende Verbindungen warten und andere ungewöhnliche Dinge.
Perform all Tiger checks on system	Diese Option führt alle oben beschriebenen Tests auf den Zielen aus.
Analyze Syslog-Files for security incidents	Zusätzlich können Sie durch Aktivieren dieser Option dafür sorgen, dass auch die Syslog-Dateien auf sicherheitskritische Einträge hin überprüft werden.
Logwatch-Einstellungen	
Analyze Syslogs low detail.	Mit dieser Level-Konfiguration erzeugt Logwatch stark zusammengefasste, wenig detaillierte Meldungen.
Analyze Syslogs medium detail.	Wenn Sie sich für diesen Level entscheiden, erzeugt Logwatch Meldungen mit einem mittleren Detailgrad.
Analyze Syslogs high detail.	Beim Level High werden ausführliche Meldungen mit dem niedrigsten Grad der Zusammenfassung erzeugt.
Fetch hardware MB sensors	Wenn Sie zusätzlich diese Option aktivieren, holt sich Logwatch auch die Daten des MB-Sensors.
Execute John the Ripper to find weak user passwords	Auch diese Option kann zusätzlich aktiviert werden. Durch Aktivieren wird der Passwort-Cracker John the Ripper aktiviert.
John-Einstellungen	
Fast crack	John kann in mehreren Modi ausgeführt werden. In diesem Modus testet John lediglich den Benutzernamen und davon abgeleitete Wörter gegen die Hash-Werte der Passwörter.

Dictionary mode	In diesem Modus werden alle Wörter des installierten Wörterbuchs genutzt, um die Hashwerte der Passwörter anzugreifen.
Full crack mode	Dieser langsamste Modus testet alle Wörter des Wörterbuchs sowie daraus generierte Variationen dieser Wörter gegen die Benutzerpasswörter.
Execute Ovaldi for scanning OVAL described issues	OVAL (Open Vulnerability and Assessment Language) ist ein Standard, der auch dazu genutzt werden kann, bekannte Sicherheitslücken und Tests zu beschreiben, mit denen festgestellt werden kann, ob diese Sicherheitslücke auf einem Zielsystem existiert. Mithilfe von Ovaldi können Sie diese Möglichkeit nutzen. Für weitere Informationen sei auf die Oval-Projektsite verwiesen (*http://oval.mitre.org*).



Der zweite Teil der SLAD-Konfiguration

Ovaldi-Berichteinstellungen	
Text	Standardmäßig werden die Ovaldi-Berichte im Textformat ausgegeben.
HTML	Alternativ ist die Ausgabe im HTML-Format möglich.
Analyse SNMP-Traps collected by snmptrap	In Ovaldi können auch die Daten, die snmptrap gesammelt hat, verarbeitet werden und in Berichte einfließen.
Fetch Snort-Events from Snort-Database	Sollen auch die Snort-Ereignisse aus der Snort-Datenbank in Ovaldi berücksichtigt werden, aktivieren Sie diese Option.
Execute ssh vulkey to detect unsecure SSH RSA and DSA key from broken Debian OpenSSL pt	Mit dieser Option kann zusätzlich der SSH-Vulkey ausgeführt werden.
Execute ChkRootKit to find installed rootkits	SLAD kann schließlich auch noch ChkRootKit ausführen, um auf den Zielen installierte Rootkits zu identifizieren.

4.2.21 SMTP settings

Mit den *SMTP settings* legen Sie verschiedene Einstellungen für das Testen von SMTP-Servern fest:

Option	Beschreibung
Third party domain	Beim Testen von Servern versucht OpenVAS, Nachrichten über den Zielrechner zu versenden. Hier gibt man die Domain Dritter an.
From address	Hier gibt man die Adresse des Senders an, der über den Ziel-SMTP-Server Nachrichten zu versenden sucht.
To address	Hier legt man das Ziel der zu versendenden Testnachrichten fest.

4.2.22 Snmpwalk

Auch die SNMP-spezifischen Funktionen von OpenVAS sind recht überschaubar. Hier stehen Ihnen lediglich drei Einstellungen zur Verfügung:

Option	Beschreibung
Community name	Wenn Sie den Community-Namen der Laufwerke kennen, können Sie deren Bezeichnung in diesem Eingabefeld hinterlegen.
SNMP protocol	Hier bestimmen Sie, ob die Protokollversion 1 oder 2c verwendet wird.
SNMP transport layer	Hier bestimmen Sie, ob TCP oder UDP im SNMP-Transportlayer verwendet werden.

4.2.23 SSL Cipher Setting

In diesem Bereich können Sie zulässige Chiffren aktivieren, die für die SSL-Verschlüsselung verwendet werden dürfen.

4.2.24 strobe

Bei strobe handelt es sich um ein Sicherheitswerkzeug, das alle TCP-Ports eines oder auch mehrerer Hosts bzgl. der Bandbreiten- und Ressourcennutzung lokalisiert und beschreibt. Sie können auf dem zugehörigen Formular insbesondere den Timeout-Wert und die Anzahl an Verbindungen bestimmen.

4.2.25 W3af

Das W3af (*http://w3af.sourceforge.net*, Web Application Attack and Audit Framework) hat die Aufgabe, Schwachstellen in Web-Applikationen aufzuspüren und auszunutzen. Wie bei OpenVAS kommen dabei Plug-ins zum Einsatz, die beispielsweise SQL-Injection- und Cross-Site-Scripting-Schwachstellen erkennen. Die aktuelle Version besitzt über 130 Skripts.

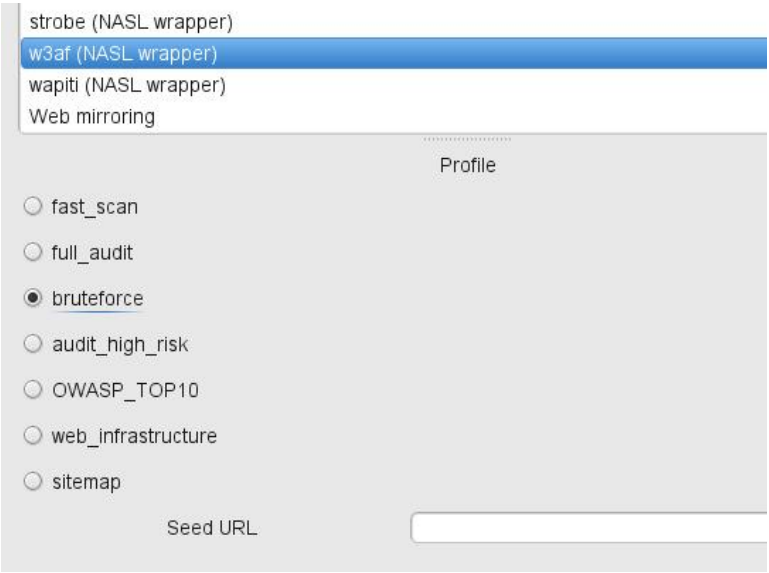
Diese Plug-ins finden die URLs, erkennen die Verwundbarkeit und sind in der Lage, diese Lücken auszunutzen. Es gibt eine weitere Gemeinsamkeit mit OpenVAS: Die Plug-ins sind in Gruppen eingeteilt. Die drei wichtigsten sind die folgenden:

- discovery

- audit
- exploit

Außerdem gibt es folgende weitere Gruppen:

- grep
- output
- mangle
- bruteforce
- evasion



Die Wahl des w3af-Profils.

Die w3af-Konfiguration stellt Ihnen verschiedene Scan-Profile zur Auswahl.

4.2.26 wapiti

Mit wapiti (*http://wapiti.sourceforge.net*) ist ein weiterer Spezialist für die Auditierung von Web-Applikationen in OpenVAS integriert. Für die Konfiguration dieses Tools steht Ihnen lediglich eine Anpassung zur Verfügung: Mit Nice begrenzen Sie das Lesen von URLs mit dem gleichen Muster. Das schützt Sie vor Endlosschleifen. Geben Sie einen Wert größer als null ein.

4.2.27 Web Mirroring

Während der HTTP-Tests versucht OpenVAS, Seiten des Ziels zu spiegeln. Hierfür stehen zwei Optionen zur Verfügung:

Option	Beschreibung
Number of pages to mirror	Hier legt man die Anzahl an Seiten fest, die OpenVAS zu spiegeln versucht.
Start page	Hier legt man den Pfad fest, von dem aus sich OpenVAS am Spiegeln versucht.

Damit kennen Sie alle erweiterten Einstellungen der OpenVAS-Clients.

4.3 Manuelle Anpassungen des Profils

Es ist in den voranstehenden Abschnitten mehrfach angeklungen: Die Funktionalität der OpenVAS-Clients variiert. Die vom OpenVAS-Server und seinen Vorgängern bereitgestellte Funktionalität hat weit mehr zu bieten, als es die meisten Clients erahnen lassen.

Dennoch müssen Sie keineswegs auf die eine oder andere Konfiguration verzichten oder einen zusätzlichen Client installieren, der die Anpassungen möglich macht.

Die Lösung ist einfach: Speichern Sie einfach Ihre aktuelle Scan-Konfiguration und nehmen Sie alle weiteren Änderungen an der Scan-Konfigurationsdatei selbst vor. Die Konfiguration für einen Bereich kann einfach über das Scope-Menü exportiert und dann mit einem beliebigen Editor bearbeitet werden. Sie kann mehrere Hundert KB groß sein.

Wie Sie nachstehendem Auszug entnehmen können, finden Sie in jeder Zeile eine Konfiguration. Die einzelnen Konfigurationsbereiche beginnen immer mit der Bezeichnung *begin* und *end*, beispielsweise *end(SERVER_PREFS)* für das Ende der Server- und *begin(PLUGINS_PREFS)* für die Konfiguration der Plug-in-Einstellungen.

Hier ein Auszug aus einer Konfiguration:

```
# This file was automatically created by OpenVAS-Client
trusted_ca = cacert.pem
hide_toolbar = no
hide_msglog = no
auto_enable_new_plugins = yes
targets = 192.168.1.11
use_client_cert = no
openvassd_port = 9390
openvassd_user = admin
paranoia_level = 1
tree_autoexpand = yes
sort_order = 0
protocol_version = 0
cache_plugin_information = yes
reports_use_plugin_cache = yes
```

```
scopes_load_plugin_cache_immediately = yes
report_plugin_details_in_pdf = yes
show_nvt_name_and_oid = yes
name = Windows XP Notebook

begin(SCANNER_SET)
 1.3.6.1.4.1.25623.1.0.10180 = yes
 1.3.6.1.4.1.25623.1.0.10278 = no
 1.3.6.1.4.1.25623.1.0.10331 = no
 1.3.6.1.4.1.25623.1.0.10335 = yes
 1.3.6.1.4.1.25623.1.0.10841 = no
 1.3.6.1.4.1.25623.1.0.10336 = no
 1.3.6.1.4.1.25623.1.0.10796 = no
 1.3.6.1.4.1.25623.1.0.11219 = no
 1.3.6.1.4.1.25623.1.0.14259 = no
 1.3.6.1.4.1.25623.1.0.14272 = no
 1.3.6.1.4.1.25623.1.0.14274 = no
 1.3.6.1.4.1.25623.1.0.14663 = no
 1.3.6.1.4.1.25623.1.0.100315 = no
 1.3.6.1.4.1.25623.1.0.80002 = no
 1.3.6.1.4.1.25623.1.0.80001 = no
 1.3.6.1.4.1.25623.1.0.80000 = no
 1.3.6.1.4.1.25623.1.0.80009 = no
 1.3.6.1.4.1.25623.1.0.11840 = yes
end(SCANNER_SET)

begin(SERVER_PREFS)
 max_hosts = 20
 max_checks = 4
 plugins_timeout = 320
```

```
 cgi_path = /cgi-bin:/scripts
 port_range = default
 auto_enable_dependencies = yes
 silent_dependencies = no
 host_expansion = ip
 ping_hosts = no
 reverse_lookup = no
 optimize_test = no
 safe_checks = no
 use_mac_addr = no
 unscanned_closed = no
 save_knowledge_base = yes
 only_test_hosts_whose_kb_we_dont_have = no
 only_test_hosts_whose_kb_we_have = no
 kb_restore = yes
 kb_dont_replay_scanners = no
 kb_dont_replay_info_gathering = no
 kb_dont_replay_attacks = no
 kb_dont_replay_denials = no
 kb_max_age = 864000
 cache_folder = /usr/local/var/cache/openvas
 include_folders = /usr/local/lib/openvas/plugins
 log_whole_attack = no
 checks_read_timeout = 5
 non_simult_ports = 139, 445
 slice_network_addresses = no
 nasl_no_signature_check = yes
end(SERVER_PREFS)

begin(CLIENTSIDE_USERRULES)
```

```
end(CLIENTSIDE_USERRULES)


begin(PLUGINS_PREFS)
 Services[entry]:Number of connections done in parallel : = 6
 Services[entry]:Network connection timeout : = 5
 Services[entry]:Network read/write timeout : = 5
 Services[entry]:Wrapped service read timeout : = 2
 Services[file]:SSL certificate : =
 Services[file]:SSL private key : =
 Services[password]:PEM password : =
 Services[file]:CA file : =
 Services[radio]:Test SSL based services = Known SSL
ports;All;None
 Login configurations[entry]:HTTP account : =
 Login configurations[password]:HTTP password (sent in clear)
: =
 Login configurations[entry]:NNTP account : =
 Login configurations[password]:NNTP password (sent in clear)
: =
 Login configurations[entry]:FTP account : = anonymous
 Login configurations[password]:FTP password (sent in clear)
: = openvas@openvas.org
 Login configurations[entry]:FTP writeable directory : =
/incoming
 Login configurations[entry]:POP2 account : =
 Login configurations[password]:POP2 password (sent in clear)
: =
 Login configurations[entry]:POP3 account : =
 Login configurations[password]:POP3 password (sent in clear)
: =
 Login configurations[entry]:IMAP account : =
 Login configurations[password]:IMAP password (sent in clear)
: =
```

```
 Login configurations[checkbox]:Never send SMB credentials in
clear text = yes

 Login configurations[checkbox]:Only use NTLMv2 = no

 Login configurations[checkbox]:NTLMSSP = no

 Global variable settings[checkbox]:Enable CGI scanning = yes

 Global variable settings[radio]:Network type = Mixed (use
RFC 1918); Public WAN (Internet);Private LAN

 Global variable settings[checkbox]:Enable experimental
scripts = no

 Global variable settings[checkbox]:Thorough tests (slow) =
no

 Global variable settings[radio]:Report verbosity = Nor-
mal;Verbose;Quiet

 Global variable settings[radio]:Report paranoia = Nor-
mal;Paranoid (more false alarms);Avoid false alarms

 Global variable settings[radio]:Log verbosity = Nor-
mal;Debug;Verbose;Quiet

 Global variable settings[entry]:Debug level = 0

 Global variable settings[entry]:HTTP User-Agent = Mozil-
la/4.0 (compatible; MSIE 6.0; Windows NT 5.0)

 Password cracking (NASL wrappers common op-
tions)[file]:Logins file : =

 Password cracking (NASL wrappers common op-
tions)[file]:Passwords file : =

 Password cracking (NASL wrappers common op-
tions)[entry]:Number of parallel tasks : = 16

 Password cracking (NASL wrappers common op-
tions)[entry]:Timeout (in seconds) : = 30

 Password cracking (NASL wrappers common op-
tions)[checkbox]:Try empty passwords = no

 Password cracking (NASL wrappers common op-
tions)[checkbox]:Try login as password = no

 Password cracking (NASL wrappers common op-
tions)[checkbox]:Exit as soon as an account is found = no

 Password cracking (NASL wrappers common op-
tions)[checkbox]:Add accounts found by other plugins to login
file = yes
```

```
 w3af (NASL wrapper)[radio]:Profile =
fast_scan;sitemap;web_infrastructure;OWASP_TOP10;audit_high_r
isk;bruteforce;full_audit

 NIDS evasion[radio]:TCP evasion technique = none;short
ttl;injection;split

 NIDS evasion[checkbox]:Send fake RST when establishing a TCP
connection = no

 Nikto (NASL wrapper)[checkbox]:Force scan even without 404s
= no

 CPE-based Policy Check[entry]:Single CPE = cpe:/

 CPE-based Policy Check[file]:CPE List =

 CPE-based Policy Check[radio]:Severity = High;Low;Medium

 CPE-based Policy Check[radio]:Severity upon =
present;missing

 3com switch2hub[entry]:Network interface on OpenVAS box
(used for scanning): =

 3com switch2hub[entry]:Fake IP (alive and on same subnet as
scanner): =

 3com switch2hub[entry]:Number of packets: = 1000000

 Web mirroring[entry]:Number of pages to mirror : = 200

 Web mirroring[entry]:Start page : = /

 LDAPsearch[entry]:Timeout value = 3

 LDAPsearch[entry]:Buffersize = 500

 SLAD Run[checkbox]:Execute Tripwire HIDS to check system's
file integrity (Linux only) = no

 SLAD Run[checkbox]:Execute ClamAV to search for virus-
infected files (Linux only) = no

 SLAD Run[radio]:ClamAV level = Move infected files to qua-
rantine;Remove infected files exclude archives (.zip, .tgz,
etc);Move infected files to quarantine exclude archives
(.zip, .tgz, etc);Remove infected files

 SLAD Run[checkbox]:Execute LSOF to retrieve a list of open
files (Linux only) = no

 SLAD Run[checkbox]:Execute Tiger for various checks (Linux
only) = no
```

```
 SLAD Run[radio]:tiger level = Checks user and passwd on lo-
cal system;Perform all Tiger checks on system;Check running
System and Processes;Check Systems Configuration and applica-
tions;Check Filesystem Permissions

 SLAD Run[checkbox]:Analyse Syslog-Files for security inci-
dents (Linux only) = no

 SLAD Run[radio]:syslogwatch level = Analyse SysLogs low de-
tail;Analyse SysLogs high detail;Analyse SysLogs medium de-
tail

 SLAD Run[checkbox]:fetch hardware MB sensors (Linux only) =
no

 SLAD Run[checkbox]:Execute John-the-Ripper to find weak user
passwords = no

 SLAD Run[radio]:john level = Fast-Crack;Full-Crack (very
slow);Dictionary Mode (slow)

 SLAD Run[checkbox]:Execute ovaldi for scanning OVAL de-
scribed issues = no

 SLAD Run[radio]:ovaldi report format = Text;HTML

 SLAD Run[checkbox]:Analyse SNMP-Traps collected by snmptrapd
(Linux only) = no

 SLAD Run[checkbox]:Fetch Snort-Events from the Snort MYSQL
Database (Linux only) = no

 SLAD Run[checkbox]:Execute ssh vulnkey to detect unsecure
SSH RSA and DSA keys from broken Debian OpenSSL pkt (Linux
only) = no

 SLAD Run[checkbox]:Execute ChkRootKit to find installed
rootkits (Linux only) = no

 SSL Cipher Settings[checkbox]:List SSL Supported Ciphers =
no

 Options for Local Security Checks[checkbox]:Also use 'find'
command to search for Applications = yes

 Options for Local Security Checks[checkbox]:Descend directo-
ries on other filesystem (don't add -xdev to find) = yes

 SSH Authorization[sshlogin]:Keys: = ignored

 SSH Authorization[checkbox]:Use per-target login information
= no

 SSH Authorization[entry]:SSH login name: = sshovas
```

```
 SSH Authorization[password]:SSH password (unsafe!): =

 SSH Authorization[file]:SSH public key: =

 SSH Authorization[file]:SSH private key: =

 SSH Authorization[password]:SSH key passphrase: =

 Misc information on News server[entry]:From address : =
OpenVAS <listme@listme.dsbl.org>

 Misc information on News server[entry]:Test group name regex
: = f[a-z]\.tests?

 Misc information on News server[entry]:Max crosspost : = 7

 Misc information on News server[checkbox]:Local distribution
= yes

 Misc information on News server[checkbox]:No archive = no

 Availability of scanner helper tools[checkbox]:Perform tool
check = yes

 Availability of scanner helper tools[checkbox]:Silent tool
check = yes

 HTTP login page[entry]:Login page : = /

 HTTP login page[entry]:Login form : =

 HTTP login page[entry]:Login form fields : = us-
er=%USER%&pass=%PASS%

 HTTP NIDS evasion[entry]:HTTP User-Agent =

 HTTP NIDS evasion[checkbox]:Use HTTP HEAD instead of GET =
no

 HTTP NIDS evasion[radio]:URL encoding = none;Incorrect UTF-
8; UTF-16 (MS %u);UTF-16 (double byte);Hex

 HTTP NIDS evasion[radio]:Absolute URI type =
none;http;gopher;file

 HTTP NIDS evasion[radio]:Absolute URI host = none;random
IP;random name;host IP;host name

 HTTP NIDS evasion[checkbox]:Double slashes = no

 HTTP NIDS evasion[radio]:Reverse traversal = none;Long
URL;Basic

 HTTP NIDS evasion[checkbox]:Self-reference directories = no

 HTTP NIDS evasion[checkbox]:Premature request ending = no
```

```
 HTTP NIDS evasion[checkbox]:CGI.pm semicolon separator = no
 HTTP NIDS evasion[checkbox]:Parameter hiding = no
 HTTP NIDS evasion[checkbox]:Dos/Windows syntax = no
 HTTP NIDS evasion[checkbox]:Null method = no
 HTTP NIDS evasion[checkbox]:TAB separator = no
 HTTP NIDS evasion[checkbox]:HTTP/0.9 requests = no
 HTTP NIDS evasion[entry]:Force protocol string : =
 HTTP NIDS evasion[checkbox]:Random case sensitivity (Nikto
only) = no
 wapiti (NASL wrapper)[entry]:Nice =
 SMB Authorization[entry]:SMB login: =
 SMB Authorization[password]:SMB password: =
 SMB Authorization[entry]:SMB domain (optional): =
 SMTP settings[entry]:Third party domain : = example.com
 SMTP settings[entry]:From address : = nobody@example.com
 SMTP settings[entry]:To address : = postmas-
ter@[AUTO_REPLACED_IP]
 Search in LDAP, Users with conf. LogonHours[entry]:Testuser
Common Name = CN
 Search in LDAP, Users with conf. LogonHours[entry]:Testuser
Organization Unit = OU
 IT-Grundschutz, 11. EL[radio]:Berichtformat = Text;Text und
Tabellarisch;Tabellarisch
 Compliance Tests[checkbox]:Launch IT-Grundschutz (10. EL) =
yes
 Compliance Tests[checkbox]:Launch IT-Grundschutz (11. EL) =
yes
 Compliance Tests[checkbox]:Verbose IT-Grundschutz results =
yes
 IT-Grundschutz, 10. EL[radio]:Berichtformat = Text;Text und
Tabellarisch;Tabellarisch
 strobe (NASL wrapper)[entry]:Strobe timeout =
 strobe (NASL wrapper)[entry]:Strobe number of sockets in
parallel =
```

```
 strobe (NASL wrapper)[entry]:Strobe local port to bind out-
going requests =

 strobe (NASL wrapper)[checkbox]:Disable usage of getpeername
= no

 ike-scan (NASL wrapper)[entry]:Source port number = 500

 ike-scan (NASL wrapper)[entry]:Destination port number = 500

 ike-scan (NASL wrapper)[checkbox]:Enable Aggressive Mode =
yes

 ike-scan (NASL wrapper)[checkbox]:Enable Main Mode = no

 ike-scan (NASL wrapper)[checkbox]:Enable fingerprint using
Aggressive Mode = no

 ike-scan (NASL wrapper)[checkbox]:Enable fingerprint using
Main Mode = no

 ike-scan (NASL wrapper)[entry]:Group names = vpn

 ike-scan (NASL wrapper)[entry]:Encryption algorithms =
1,2,3,4,5,6,7/128,7/196,7/256,8

 ike-scan (NASL wrapper)[entry]:Hash algorithms = 1,2,3,4,5,6

 ike-scan (NASL wrapper)[entry]:Authentication methods =
1,2,3,4,5,6,7,8,64221,65001

 ike-scan (NASL wrapper)[entry]:Diffie-Hellman groups =
1,2,3,4,5

 ike-scan (NASL wrapper)[entry]:Maximum retry = 3

 ike-scan (NASL wrapper)[entry]:Maximum timeout =

 pnscan (NASL wrapper)[entry]:Pnscan Timeout =

 pnscan (NASL wrapper)[entry]:Pnscan Concurrent worker
threads =

 amap (NASL wrapper)[file]:File containing machine readable
results : =

 amap (NASL wrapper)[radio]:Mode = Port scan only;Just grab
banners;Map applications

 amap (NASL wrapper)[checkbox]:Quicker = yes

 amap (NASL wrapper)[checkbox]:UDP scan (disabled in
safe_checks) = no

 amap (NASL wrapper)[checkbox]:SSL (disabled in safe_checks)
= yes
```

```
 amap (NASL wrapper)[checkbox]:RPC (disabled in safe_checks)
= yes

 amap (NASL wrapper)[entry]:Parallel  tasks =

 amap (NASL wrapper)[entry]:Connection retries =

 amap (NASL wrapper)[entry]:Connection timeout =

 amap (NASL wrapper)[entry]:Read timeout =

 snmpwalk 'scanner'[entry]:Community name : = public

 snmpwalk 'scanner'[radio]:SNMP protocol : = 1;2c

 snmpwalk 'scanner'[radio]:SNMP transport layer : = udp;tcp

 snmpwalk 'scanner'[entry]:TCP/UDP port : =

 snmpwalk 'scanner'[entry]:Number of retries : =

 snmpwalk 'scanner'[entry]:Timeout between retries : =

 Nmap (NASL wrapper)[radio]:TCP scanning technique : = con-
nect();Null scan;Xmas Tree scan;FIN scan;SYN scan

 Nmap (NASL wrapper)[checkbox]:UDP port scan = no

 Nmap (NASL wrapper)[checkbox]:Service scan = no

 Nmap (NASL wrapper)[checkbox]:RPC port scan = no

 Nmap (NASL wrapper)[checkbox]:Identify the remote OS = no

 Nmap (NASL wrapper)[checkbox]:Use hidden option to identify
the remote OS = no

 Nmap (NASL wrapper)[checkbox]:Fragment IP packets (bypasses
firewalls) = no

 Nmap (NASL wrapper)[checkbox]:Get Identd info = no

 Nmap (NASL wrapper)[checkbox]:Do not randomize the  order
in  which ports are scanned = no

 Nmap (NASL wrapper)[entry]:Source port : =

 Nmap (NASL wrapper)[radio]:Timing policy : = Auto (openvas
specif-
ic!);Custom;Paranoid;Sneaky;Polite;Aggressive;Insane;Normal

 Nmap (NASL wrapper)[entry]:Host Timeout (ms) : =

 Nmap (NASL wrapper)[entry]:Min RTT Timeout (ms) : =

 Nmap (NASL wrapper)[entry]:Max RTT Timeout (ms) : =

 Nmap (NASL wrapper)[entry]:Initial RTT timeout (ms) : =
```

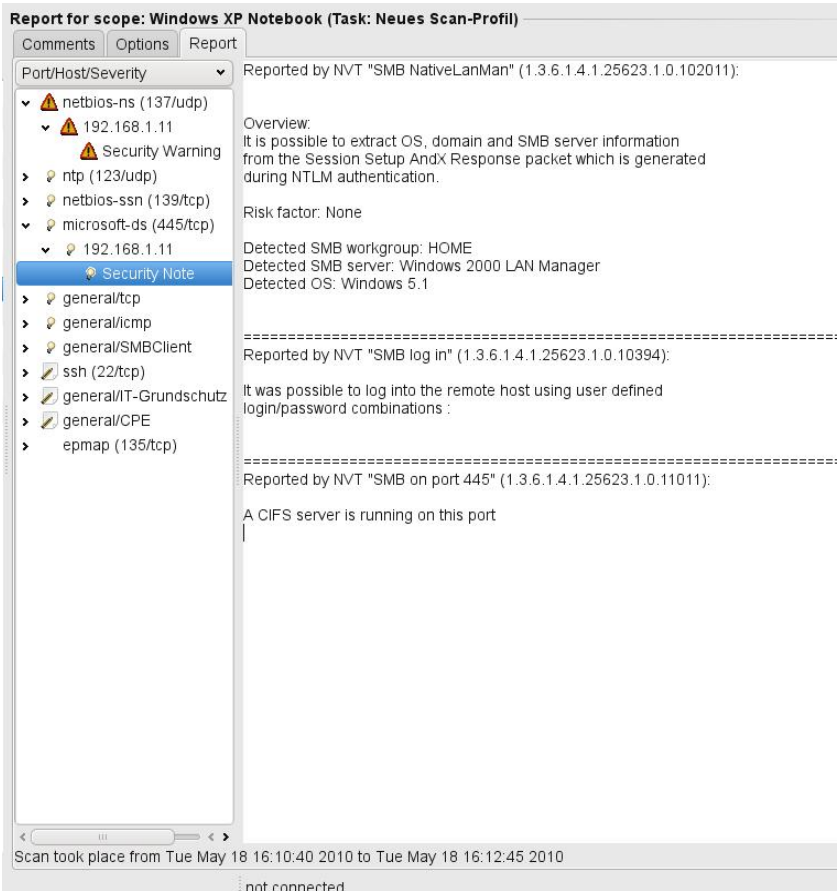
```
 Nmap (NASL wrapper)[entry]:Ports scanned in parallel (max) =
 Nmap (NASL wrapper)[entry]:Ports scanned in parallel (min) =
 Nmap (NASL wrapper)[entry]:Minimum wait between probes (ms)
=
 Nmap (NASL wrapper)[file]:File containing grepable results :
=
 Nmap (NASL wrapper)[checkbox]:Do not scan targets not in the
file = no
 Nmap (NASL wrapper)[checkbox]:Run dangerous port scans even
if safe checks are set = no
 portbunny (NASL wrapper)[checkbox]:Wait longer for triggers
to return = no
 Ping Host[checkbox]:Report about unreachable Hosts = no
 Ping Host[checkbox]:Mark unreachble Hosts as dead (not scan-
ning) = no
 w3af (NASL wrapper)[entry]:Seed URL =
end(PLUGINS_PREFS)


begin(PLUGIN_SET)
 1.3.6.1.4.1.25623.1.0.10330 = yes
 1.3.6.1.4.1.25623.1.0.66557 = no
 1.3.6.1.4.1.25623.1.0.63444 = yes
 1.3.6.1.4.1.25623.1.0.53249 = no
 1.3.6.1.4.1.25623.1.0.63962 = no
 1.3.6.1.4.1.25623.1.0.63855 = no
 1.3.6.1.4.1.25623.1.0.61182 = no
 1.3.6.1.4.1.25623.1.0.861227 = no
 1.3.6.1.4.1.25623.1.0.860482 = no
 1.3.6.1.4.1.25623.1.0.870173 = no
 1.3.6.1.4.1.25623.1.0.861874 = no
 1.3.6.1.4.1.25623.1.0.861044 = no
 ...
 ...
```

```
 ...
 1.3.6.1.4.1.25623.1.0.100644 = yes
 1.3.6.1.4.1.25623.1.0.861950 = yes
 1.3.6.1.4.1.25623.1.0.861941 = yes
 1.3.6.1.4.1.25623.1.0.861938 = yes
end(PLUGIN_SET)

begin(SERVER_INFO)
 server_info_openvassd_version = 3.0.2
 server_info_libnasl_version = 3.0.5.
 server_info_libnessus_version = 3.0.5
 server_info_thread_manager = fork
 server_info_os = Linux
 server_info_os_version = 2.6.31.5-0.1-desktop
end(SERVER_INFO)
```

Nachdem Sie die Änderungen vorgenommen haben, importieren Sie einfach die geänderte Datei oder aber überschreiben Sie die bestehende.

5 Berichte verstehen und interpretieren

Nachdem Sie Ihre ersten Scans durchgeführt und womöglich mit der einen oder anderen Scan-Konfiguration gespielt haben, sehen Sie im Register *Report* die Ergebnisse Ihrer Tests.



Das OpenVAS-Client 3.0.0 gibt einen ersten Bericht aus.

Da es mit der Durchführung der eigentlichen Tests natürlich nicht getan ist, müssen Sie als Nächstes die Ergebnisse, die OpenVAS ausgibt, analysieren, verstehen und korrekt auswerten. Dazu ist es zunächst einmal erforderlich, dass Sie den Report-Viewer kennen und die dort präsentierten Daten interpretieren lernen.

5.1 Der Bericht-Viewer

Schauen wir uns zunächst den im OpenVAS-Client integrierten Report-Viewer an. Nach der Ausführung der ersten Scan-Vorgänge werden den einzelnen Bereichen in der linken Hierarchie die Berichte untergeordnet. Mit einem Klick auf einen Bericht wird im rechten Fensterbereich der Bericht-Viewer geöffnet.

Die Berichte werden chronologisch aufgeführt, wobei jedem Bericht eine Bezeichnung in folgendem Format zugewiesen wird:

```
Report Datum-Uhrzeit
```

Es folgt eine tabellarische Übersicht mit der Risikobewertung (niedrig, mittel und hoch) der gefundenen Schwachstellen und der Anzahl an Log-Einträgen.

Mit einem Klick auf eine Berichtbezeichnung öffnet der OpenVAS-Client im rechten Fensterbereich die Berichtdetails auf dem Report-Register. Dieses ist zweigeteilt: links finden Sie eine Liste der gefundenen Schwachstellen bzw. Risiken, rechts die dazugehörigen Informationen. Über die linke Spalte navigieren Sie zu den einzelnen Testergebnissen.

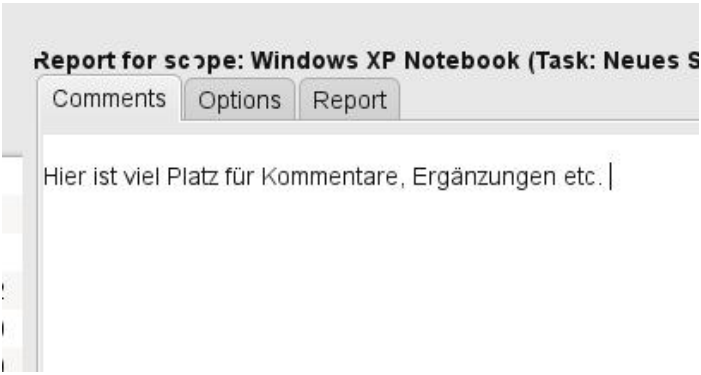
Der OpenVAS-Client verwendet für die Kennzeichnung der Risikobewertung vier verschiedene Icons:

Icon	Kurzinfo
	Das Einbahnstraßenschild weist auf eine Schwachstelle mit hohem Sicherheitsrisiko hin.
	Das Warnschild „Andere Warnung“, das Sie ebenfalls aus dem Straßenverkehr kennen, allerdings mit orangem Hintergrund, weist auf ein mittleres Sicherheitsrisiko hin.
	Die Glühbirne zeigt ein schwaches Sicherheitsrisiko an.
	Das Bericht-Icon weist auf nützliche Sicherheitsinformationen hin.

Im Unterschied zu seinem Vorläufer kommt der Bericht-Viewer leider nicht mit einer Suchfunktion daher, die Sie bei der Suche nach bestimmten Informationen in einem Bericht unterstützen würde.

Dafür hat der OpenVAS-Client zwei andere Besonderheiten zu bieten: Sie können die Berichte um Kommentare ergänzen und Sie können den Schweregrad manuell nachbearbeiten.

Um einem Bericht Kommentare hinzuzufügen, wechseln Sie einfach auf das Register *Comments* und geben dort die gewünschten Informationen an. Fertig. Wenn Sie die Kommentarfunktion intensiv nutzen, sollten Sie sich das gsi-Modul genauer ansehen. Dort stehen Ihnen weit mehr Kommentarmöglichkeiten zur Verfügung.



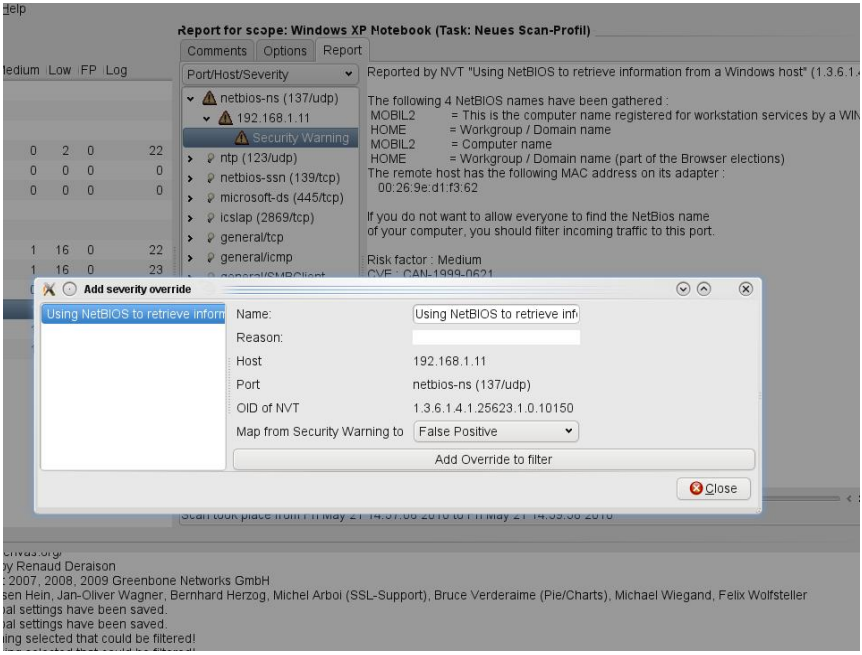
Im *Comments*-Register hinterlegen Sie Ihre Kommentare.

Die zweite bereits erwähnte Besonderheit der Berichtfunktion erlaubt das Bearbeiten der Severities, also der Einschätzung des Schweregrads. Dazu navigieren Sie im Bericht-Viewer zu einer Meldung und führen mit der rechten Maustaste den Befehl *Severities ...* aus.

In dem zugehörigen Dialog wird zunächst im linken Bereich die Bezeichnung der Schwachstelle aufgeführt. In nachstehendem Beispiel ist es *Using NetBIOS to retrieve information from a Windows host*.

Um den Status einer Ausgabe zu ändern, geben Sie im Eingabefeld *Reason* eine Begründung für die Änderung an. Es folgen die Detailinformation zum Hostnamen bzw. der IP-Adresse, dem Port und der OID. Mit dem Auswahlmenü *Map from Security Warning to* können Sie den Status in einen der drei bekannten Risikobewertungen ändern oder diese als *False Positive* kennzeichnen. Auf Letztere kommen wir weiter unten noch zu sprechen.

Die Statusänderung schließen Sie mit einem Klick auf *Add Override to filter* ab.



Das Bearbeiten des Schweregrads einer Warnung.

Der im OpenVAS integrierte Viewer genügt meines Erachtens eigentlich nur, um sich einen ersten Eindruck von den Testergebnissen zu verschaffen. Wenn Sie den Bericht beispielsweise nach HTML exportieren, so können Sie diesen deutlich besser auswerten und weiterverarbeiten.

Damit Sie einen ersten Eindruck davon bekommen, was Sie alles an Ergebnissen erwartet, hier die Ausgabe zu einer mittelschweren Warnung:

```
Reported by NVT "Using NetBIOS to retrieve information from a
Windows host" (1.3.6.1.4.1.25623.1.0.10150):


The following 4 NetBIOS names have been gathered :

 MOBIL2          = This is the computer name registered for
workstation services by a WINS client.
```

```
 HOME            = Workgroup / Domain name

 MOBIL2          = Computer name

 HOME            = Workgroup / Domain name (part of the
Browser elections)

The remote host has the following MAC address on its adapter
:

   00:26:9e:d1:f3:62


If you do not want to allow everyone to find the NetBios name

of your computer, you should filter incoming traffic to this
port.


Risk factor : Medium

CVE : CAN-1999-0621
```

Wie Sie voranstehender Ausgabe entnehmen können, gibt OpenVAS zunächst die Bezeichnung des Tests bzw. der Schwachstelle zurück, gefolgt von der OID.

Es folgt eine Kurzinfo, in der die Schwachstelle zusammengefasst ist. Sofern das Skript auch eine Lösung kennt, wird diese als Nächstes präsentiert.

Auf die Hinweise zur Lösung eines Problems folgen in der Regel eine Risikobewertung sowie weitere Referenzen. Gibt das Skript irgendwelche Ausgaben zurück, so finden Sie diese ebenfalls in der Berichtausgabe.

Die Reihenfolge der jeweiligen Informationen kann übrigens variieren. Außerdem müssen Sie damit rechnen, dass nicht bei allen Berichtausgaben alle oben aufgeführten Details verfügbar sind. Ob, und wenn ja, welche Informationen ausgegeben werden, ist letztlich Sache des Skript-Entwicklers.

Die möglichen Berichtinfos im Überblick

Die folgenden Informationen können Sie in einem Bericht finden:

Name: In der ersten Zeile des Berichts findet man meist eine einzeilige Bezeichnung des Problems bzw. des Testskripts. Sie lässt in der Regel erkennen, worum es sich hier handelt.

Beschreibung: Es folgt eine mehrzeilige Beschreibung der Lücke bzw. des Hinweises. Diese Informationen werden bei der Berichtgenerierung dem Testskript entnommen und in den Bericht eingefügt.

Solution: Abhängig vom Skript folgen auf die Beschreibung Lösungshinweise, wie man das Sicherheitsproblem in den Griff bekommt.

Risk factor: In der Berichtausgabe findet man meist eine Beurteilung des Risikos, das mit der jeweiligen Lücke bzw. Warnung verbunden ist. Hier findet man folgende Einschätzungen: Low, Medium, High und Critical. Die Beurteilung erfolgt durch ein von William Heinbockel geschriebenes Skript. Wichtig für den Administrator ist natürlich, dass zumindest alle kritischen Risiken beseitigt werden.

CVE: Unter Umständen sind auch CVE-IDs (common Vulnerabilities and Exposures) in der Berichtansicht enthalten. Dabei handelt es sich um eine Initiative von CERT und dem US-Ministerium für Heimatsicherheit. Durch die Angabe einer ID ist ein eindeutiger Bezug zu entsprechenden CVE- oder CAN-Artikeln hergestellt.

BID: Schließlich findet man in der Berichtausgabe gelegentlich sogenannte BugTraq-IDs (BID), die von SecurityFocus für Bugberichte vergeben werden. Über *http://www.securityfocus.com/bid/ziffernfolge* greifen Sie auf ergänzende Informationen zu.

Wenn Sie im Bericht-Viewer in den Bereich general/tcp wechseln finden Sie dort übrigens eine sehr detaillierte Zusammenfassung der Scan-Konfiguration und Aktivitäten. Hier ein Beispiel:

```
Reported by NVT "Information about the scan"
(1.3.6.1.4.1.25623.1.0.19506):


Information about this scan :


OpenVAS version : 3.0.5.
Plugin feed version : 201005191228
Type of plugin feed : OpenVAS NVT Feed
```

```
Scanner IP : 192.168.1.4
Port scanner(s) : openvas_tcp_scanner
Port range : default
Thorough tests : no
Experimental tests : no
Paranoia level : 1
Report Verbosity : 1
Safe checks : no
Max hosts : 20
Max checks : 4
Scan Start Date : 2010/5/27 11:13
Scan duration : 193 sec

================================================================

Reported by NVT "SLAD Microsoft Baseline Security Analyzer
Updates run" (1.3.6.1.4.1.25623.1.0.96063):


This script connects to SLAD on a remote host to run remote
the Microsoft Baseline Security Analyzer. To work properly,
this script requires to be provided with a valid SSH login by
means of an SSH key with passphrase if the SSH public key is
passphrase-protected, or a password to log in.


================================================================

Reported by NVT "SLAD fastjohn Run"
(1.3.6.1.4.1.25623.1.0.96061):


This script connects to SLAD on a remote host to run remote
john password scanner in fastmode. To work properly, this
script requires to be provided with a valid SSH login by
means of an SSH key with passphrase if the SSH public key is
passphrase-protected, or a password to log in.

================================================================
```

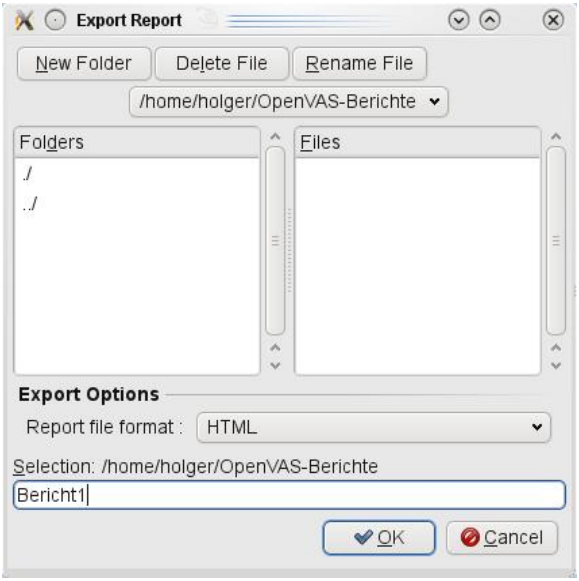
```
Reported by NVT "SLAD Microsoft Baseline Security Analyzer
ALL run" (1.3.6.1.4.1.25623.1.0.96065):


This script connects to SLAD on a remote host to run remote
the Microsoft Baseline Security Analyzer. To work properly,
this script requires to be provided with a valid SSH login by
means of an SSH key with passphrase if the SSH public key is
passphrase-protected, or a password to log in.


================================================================

Reported by NVT "SLAD Netstat -natcp run"
(1.3.6.1.4.1.25623.1.0.96066):


This script connects to SLAD on a remote host to run remote
the Microsoft Baseline Security Analyzer. To work properly,
this script requires to be provided with a valid SSH login by
means of an SSH key with passphrase if the SSH public key is
passphrase-protected, or a password to log in.


================================================================

Reported by NVT "SLAD Microsoft Baseline Security Analyzer OS
run" (1.3.6.1.4.1.25623.1.0.96064):


This script connects to SLAD on a remote host to run remote
the Microsoft Baseline Security Analyzer. To work properly,
this script requires to be provided with a valid SSH login by
means of an SSH key with passphrase if the SSH public key is
passphrase-protected, or a password to log in.


================================================================

Reported by NVT "SLAD Microsoft (R) Windows (R) Resource
Checker run" (1.3.6.1.4.1.25623.1.0.96062):


This script connects to SLAD on a remote host to run remote
the Microsoft (R) Windows (R) Resource Checker. To work prop-
erly, this script requires to be provided with a valid SSH
login by means of an SSH key with passphrase if the SSH pub-
lic key is passphrase-protected, or a password to log in.
```

```
==============================================================

Reported by NVT "SLAD Run" (1.3.6.1.4.1.25623.1.0.90002):


This script connects to SLAD on a remote host to run remote
scanners. To work properly, this script requires to be pro-
vided with a valid SSH login by means of an SSH key with
passphrase if the SSH public key is passphrase-protected, or
a password to log in.


==============================================================

Reported by NVT "SLAD Fetch Results"
(1.3.6.1.4.1.25623.1.0.90003):


This script connects to SLAD on a remote host to fetch the
result from scripts started earlier. To work properly, this
script requires to be provided with a valid SSH login by
means of an SSH key with passphrase if the SSH public key is
passphrase-protected, or a password to log in.


==============================================================

Reported by NVT "Nikto (NASL wrapper)"
(1.3.6.1.4.1.25623.1.0.14260):


Nikto could not be found in your system path. OpenVAS was
unable to execute Nikto and to perform the scan you re-
quested. Please make sure that Nikto is installed and that
nikto.pl or nikto is available in the PATH variable defined
for your environment.


==============================================================

Reported by NVT "SSH Authorization"
(1.3.6.1.4.1.25623.1.0.90022):


No port for an ssh connect was found open. Hence local secu-
rity checks might not work.
```

Leider sind die Informationen nicht sehr übersichtlich. Daher empfiehlt es sich, die Daten entweder in ein Exportformat Ihrer Wahl zu überführen oder aber den GSA (Greenbone Security Assistant) einzusetzen. Die Exportfunktion ist Thema des nächsten Abschnitts, auf den GSA kommen wir im Kapitel 6 zu sprechen.

5.2 Berichtexport

Wenn Sie einen ersten Bericht erstellt haben und diesen mit Dritten austauschen wollen, so greifen Sie zur Exportfunktion, die Ihnen das *Report*-Menü bietet.

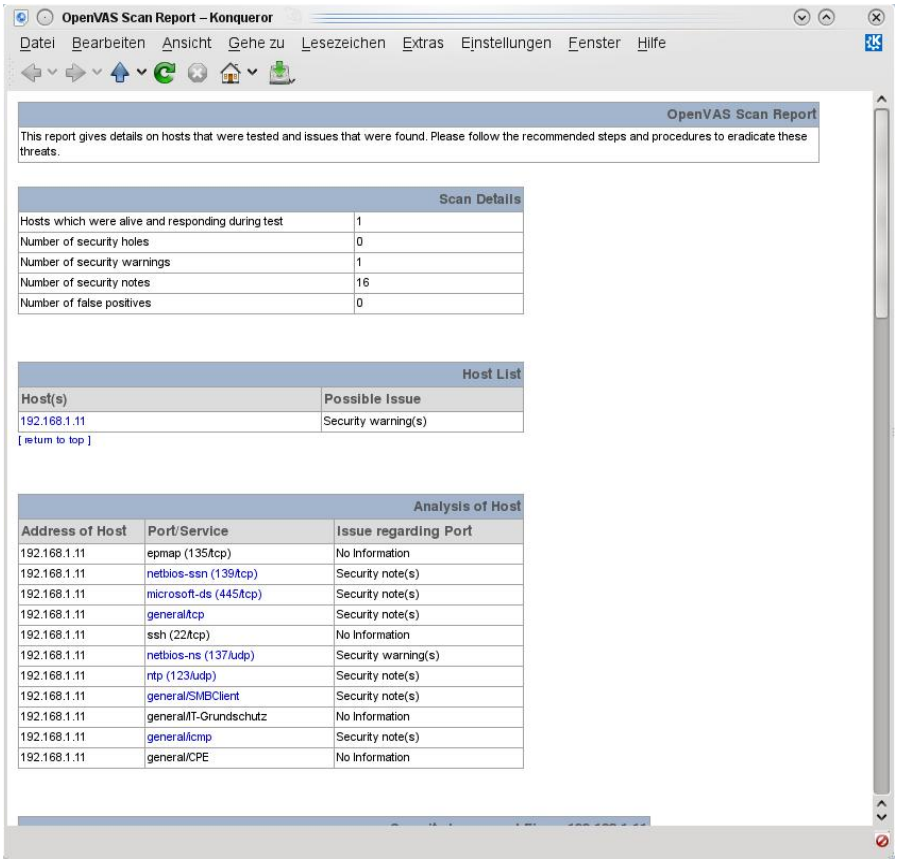


Die Exportfunktion des OpenVAS-Clients.

So können Sie die Testergebnisse mit Kollegen oder einem Service-Dienstleister diskutieren und über mögliche Schutzmaßnahmen beratschlagen.

Um einen Bericht zu sichern, öffnen Sie diesen und führen über das Menü *Report* den Befehl *Export* aus. Es meldet sich der *Export Report*-Dialog, auf dem Sie zunächst das Ziel und den Dateinamen bestimmen. Über das Auswahlmenü *Report File Format* bestimmen Sie das Zieldateiformat. Beim OpenVAS-Client haben Sie

die Wahl zwischen folgenden Formaten: NBE, XML, HTML, LaTeX, ASCII und PDF.



Ein typischer HTML-Bericht.

Nachstehende Tabelle fasst die Formate und ihre Eigenschaften zusammen.

Format	Beschreibung
NBE	Beim Nessus Backend-Format handelt es sich um eine Pipe-getrennte Textdatei, in der jede Lücke in einer eigenen Zeile berichtet wird. Dieses Format eignet sich insbesondere für das Parsen mit awk oder vergleichbaren Tools. NBE kommt außerdem bei der Übermittlung von Reports zwischen OpenVAS-Clients zum Einsatz. Das Format sieht wie folgt aus: *hostname* | *port* oder *hostname* | *port* | *skript-ID* | *typ* | *daten* Das erste Format zeigt an, dass es sich um einen offenen Port handelt. Das zweite Format ergänzt einen Sicherheitsbereich. Wie man am zweiten Format erkennen kann, können in einer NBE-Datei folgende Informationen hinterlegt werden: • Hostname: Gibt den Hostnamen oder die IP-Adresse des untersuchten Hosts an. • Port: Gibt den kritischen Port an, wobei das Format *Portnummer/Protokoll* verwendet wird, beispielsweise *www(80/tcp)*. • Skript-ID: Hinterlegt die ID des jeweiligen Test-Skripts. • Typ: Zeigt an, ob es sich um eine Info oder eine Warnung handelt. • Daten: Hier werden die Berichtinformationen hinterlegt.
XML	Mit dieser Option erzeugen Sie einen XML-basierten Bericht. Diese Variante dürfe insbesondere dann interessant sein, wenn Sie die Daten in Drittanwendungen weiterverarbeiten wollen.
HTML	Diese Option erzeugt einen einfachen HTML-basierten Bericht, der natürlich plattformübergreifend genutzt werden kann. Neben einer Zusammenfassung enthält er auch jede Menge Detailinformationen. Das Besondere an einem HTML-Bericht: Meist sind Verweise zu externen Lösungen und/oder Informationen als Hyperlinks implementiert und können direkt aus dem Bericht heraus angesteuert werden. Gleiches gilt für die Skript-Infos. Über die OID

	öffnen Sie die zugehörigen Informationen in der Skript-Datenbank.
LaTeX	Wenn Sie einen TEX-basierten Bericht bevorzugen, wählen Sie diese Exportvariante.
ASCII	Als vorletztes Format ist das Erstellen einer Textdatei.
PDF	Erzeugt aus den Berichten eine PDF-Datei. In dem PDF kommen farbige Kennzeichnungen zum Einsatz Dabei wird die Kopfzeile farbig unterlegt. Konkret werden folgende Markierungen verwendet: • Blau für unkritische Ausgaben • Orange und gelb für mittelschwere Sicherheitsrisiken • Rot für kritische Ereignisse Diese farbigen Hervorhebungen finden Sie übrigens auch beim HTML-Export.

5.3 Scanner-Logik

Neben diesem allgemeinen Background sollten Sie die Scanner-Logik verstehen. Wenn Sie wissen, wie der Scanner tickt, fällt es leichter, die Ergebnisse korrekt zu interpretieren. Wie wir bereits wissen, basieren die Scan-Vorgänge auf NASL-Skripts. Für jedes dieser Skripts führt der Scanner folgende Schritte aus:

1. Zunächst stellt der Scanner sicher, dass alle in den Abhängigkeiten der Skripts definierten Scans durchgeführt wurden.
2. Dann findet der Scanner heraus, ob der betreffende Dienst auf dem Ziel-Host überhaupt ausgeführt wird. Dazu wird die Knowledge Base konsultiert – sofern diese aktiviert ist.
3. Sofern möglich, werden Sicherheitslücken erkannt.
4. Als Nächstes versucht sich der Scanner an der Verifizierung der Sicherheitslücke. Dieser Vorgang ist von Skript zu Skript sehr unterschiedlich geregelt. Meist sendet das Skript bestimmte Werte oder Kommandos an einen Dienst und analysiert dessen Antworten.
5. Schließlich bestimmt OpenVAS auf Grundlage der zurückgegebenen Informationen, ob es sich um eine kritische Sicherheitslücke handelt. Dazu wird ein sogenannter Risikofaktor gesetzt.

Mit diesen Hintergrundinformationen sind Sie bestens für die Auswertung von Berichten vorbereitet.

5.4 Bericht interpretieren

Als Nächstes gilt es, die Informationen, die Sie einem Bericht entnehmen können, zu interpretieren. Dabei geht es darum herauszulesen, in welchen Informationen was tatsächlich enthalten ist, wie die Informationen in den Gesamtkontext der Umgebung passen und welche Schlussfolgerungen gezogen werden müssen. Auch die Frage, ob die Ausgaben die tatsächlichen Verhältnisse widerspiegeln, muss man sich stellen.

Sie sollten sich zunächst vergegenwärtigen, dass OpenVAS potenzielle Schwachstellen lediglich aufgrund eines einfachen Frage-Antwort-Spiels beurteilt. Ein umfangreicher Sicherheitscheck, der sich beispielsweise am Ausnutzen eines offenen Ports versucht, findet selten statt. OpenVAS wertet in erster Linie die Rückgaben der Hosts und der angesprochenen Dienste aus. Wollen Sie die Ausgaben von Nessus von Grund auf verstehen, so müssen Sie sich insbesondere mit NASL und dem Aufbau und der Abarbeitung von Testskripts befassen.

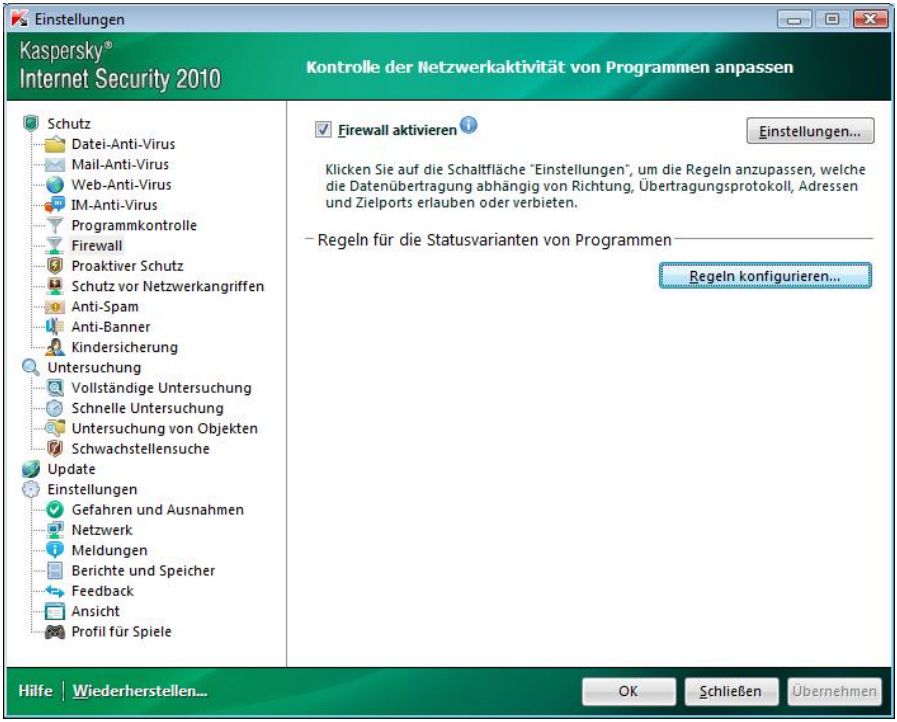
Aber auch ohne tiefgehende NASL-Kenntnisse kann man den Berichten alle notwendigen Informationen für das Verstehen und die Beseitigung eines Risikos entnehmen. Die wichtigsten Informationen sind die Zusammenfassung, die Risikoklassifizierung und die Lösungsvorschläge.

Bei der Interpretation der Berichtausgabe ist es auch wichtig, dass Sie sich vergegenwärtigen, welche Faktoren Einfluss auf die Berichte haben. Und das sind mehr, als einem manchmal lieb sind. Manche haben größeren Einfluss, andere wiederum einen recht kleinen. Leider lässt sich allgemein nicht sagen, in welchem Umfang diese Faktoren die Ergebnisse verfälschen. Wichtig ist, dass man sich dieser Einflussfaktoren bewusst ist. Die wichtigsten Faktoren:

- **Plug-in-Wahl**: Einen erheblichen Einfluss hat die Wahl der Plug-ins. Über das Nessus-Setup erfolgt die Auswahl der Plug-ins. Hier legt man neben der übergeordneten Kategorie auch Details fest. Aktiviert man beispielsweise das Plug-in Windows, das für die Analyse von Windows-basierten Systemen bestimmt ist, so kann man beispielsweise festlegen, ob das System auch die Instant Messenger von AOL und Yahoo ausführt. Über das Plug-in-Setup legen Sie natürlich fest, ob alle Plug-ins und auch solche, die dem Ziel gefährlich werden können, ausgeführt werden. Die Wahl der Plug-ins hat natürlich die direktesten Auswirkungen auf die Berichtausgabe. Tests, die nicht ausgeführt werden, können auch keine Ergebnisse liefern.

- **Plug-in-Abhängigkeiten**: Ähnlich verhält es sich mit Abhängigkeiten, die zwischen den Testskripts bestehen. In der Standardeinstellung wird ein Skript erst dann ausgeführt, wenn all seine Abhängigkeiten erfüllt sind. Mit Abhängigkeiten ist in diesem Zusammenhang gemeint, dass Skripts auf den Ergebnissen anderer Skripts aufbauen und deren Ergebnisse für ihre Ausführung voraussetzen.

  Aktivieren Sie im Plug-in-Setup die Option *Enable dependencies at runtime*, so werden während der Skriptausführung die erforderlichen Skripts ausgeführt. So haben auch die Abhängigkeiten der einzelnen Skripts untereinander ihren Einfluss auf die Berichtausgabe. Gleiches gilt auch für Filter, die man beim Einsatz der Plug-ins setzt.



Eine auf einem Windows-Client installierte Firewall erkennt einen Scan-Versuch und hat damit ebenfalls Auswirkungen auf das Ergebnis.

- **no404- und andere kritische Tests**: Es gibt übrigens auch Tests, die immer ein negatives Ergebnis liefern, so beispielsweise der Test *safe_checks*. Auch das Testskript *no404.nasl* hat Auswirkungen auf das Testergebnis. Es wird verwendet, um Webserver zu testen, ob sie bei der Ausgabe den HTTP-Code 200 OK verwenden. Gelegentlich treten bei der Ausführung des Test-Skripts Fehler auf, die sich dann auch in dem Ergebnis widerspiegeln. Im Zweifelsfall sollten Sie beim Scannen eines Webservers den zugehörigen Bericht exakt auf mögliche Fehler hin untersuchen.
- **Portscanner-Einstellungen**: Natürlich haben auch die Einstellungen und die Wahl des Portscanners einen entscheidenden Einfluss auf die Berichte. Die hierfür zuständigen Einstellungen finden Sie in den Scan-Einstellungen. Einfluss hat nicht nur die Wahl der Scanner, sondern auch deren korrekte Installation und Konfiguration. Soweit es die Scans betrifft, hat die Option *Consider unscanned ports as closed* einen erheblichen Einfluss auf den Bericht. Sie geht davon aus, dass nicht gescannte Ports sicher sind. Dass das nicht notwendigerweise der Fall sein muss, versteht sich von selbst.
- **Proxy Server, Firewalls und TCP Wrapper**: Auch diese Komponenten einer typischen Infrastruktur beeinflussen die Ergebnisse, insbesondere dann, wenn sie beispielsweise unerwünschten Datenverkehr herausfiltern oder einzelne Systeme durch eine eigene Firewall geschützt sind.
- **Weitere Faktoren**: Die Liste der (negativen) Einflussfaktoren ließe sich fortsetzen. So beeinflussen beispielsweise auch SSL-basierte Dienste, differenzielle Scans und die Funktion *Optimize the test* das Ergebnis. Nicht zu vergessen sind Bugs in den Testskripts. Zumindest den Bugs kann man weitgehend begegnen, indem man dafür sorgt, dass man immer die aktuellsten Skripts verwendet.

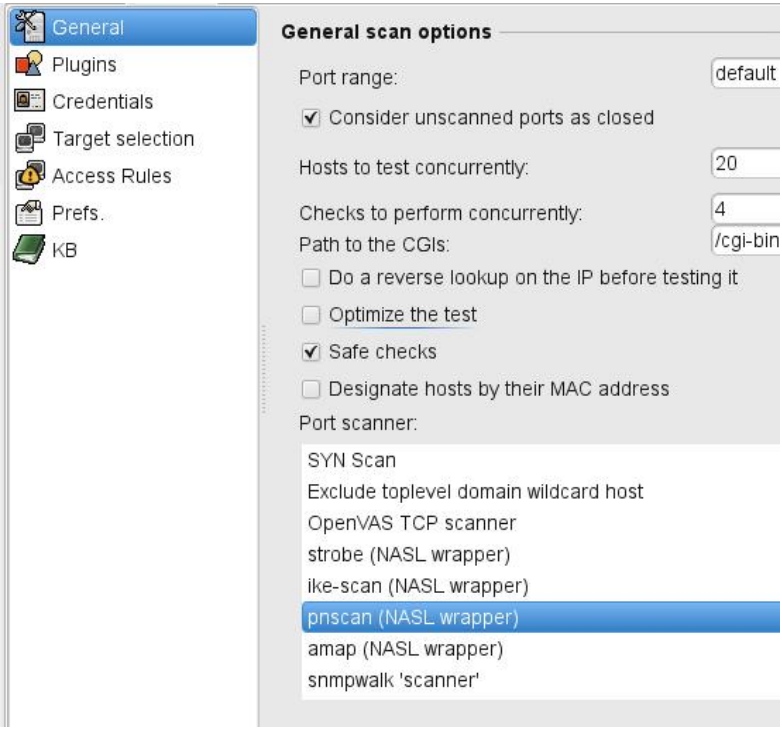
5.6 Umgang mit False Positives

Der Albtraum eines jeden Administrators sind die sogenannten *False Positives*. Dabei handelt es sich um Fehlalarme, die oft als Folge einer nicht ausreichenden Konfiguration dem Administrator das Leben schwer machen. Falschmeldungen können aber auch durch eine unvollständige Ausführung oder durch Bugs verursacht werden.

Sie können auch durch veraltete Testskripts entstehen, die eine bereits gepatchte Sicherheitslücke unter die Lupe nehmen. Für das Testen von Verwundbarkeiten eines Systems gibt es, wie bereits oben erwähnt, zwei Ansätze:

- eindringendes Scannen
- nicht-eindringendes Scannen

Im ersten Fall sendet man an den jeweiligen Dienst Daten, die die Schwachstelle ausnutzen und beispielsweise ein System zum Absturz bringen. Bei der zweiten Methode sendet man Anfragen an den Dienst, die die Schwachstelle verifizieren, den Dienst aber selbst nicht lahm legen oder auf eine andere Art beschädigen.

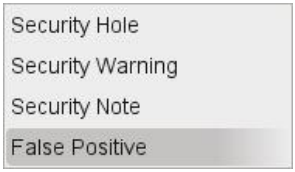


Mit der optimalen Konfiguration der Scan-Optionen begegnet man False Positives.

OpenVAS unterstützt beide Verfahren. Um die ungefährlichere Variante zu wählen, stellen Sie sicher, dass die Option *Safe checks* auf dem Register *General* der Scan-Optionen aktiviert ist. Diese Option sorgt auch dafür, dass nicht unnötige Informationen in den Berichten landen.

Für den Administrator ist es besonders schwierig, sie als solche zu identifizieren. Es dauert nicht nur lange, bis sie als solche erkannt und behoben sind, sondern sie können sogar ganze Testreihen (auch solche über einen längeren Zeitraum hinweg) infrage stellen. Also muss man Wege finden, wie man mit solchen Problemen umgeht. Auch hierfür ist OpenVAS gerüstet.

Für den Umgang mit False Positives hat der Bericht Viewer noch eine Besonderheit zu bieten. Wie wir in Kapitel 5.1 gesehen haben, können Sie im Bericht-Viewer mithilfe der Severities-Funktion den Schweregrad einer Warnung bearbeiten. Wenn Sie eine Warnung als Falschmeldung identifiziert haben, können Sie diese auch als False Positive kennzeichnen und damit aus der Warnungsübersicht nehmen. Führen Sie dazu einfach aus dem Auswahlmenü *Map from Security Warnings to* die Option *False Positive* aus.



Das Ändern des Warnungsstatus in *False Positive*.

Ein weiteres Problem kann zu Unschärfen bei der Berichtausgabe führen. Die Rede ist vom sogenannten Rauschen. Damit bezeichnet man im Zusammenhang mit Nessus technisch zwar korrekte Berichtinformationen, die aber irrelevante Daten ausgeben und es so für den Anwender erschweren, die relevanten Informationen herauszuziehen. Gerade beim Testen ganzer Subnetze mit vielen Rechnern steigt der Informationsgehalt derart, dass die echte Fehler- und Sicherheitslückensuche der sprichwörtlichen Suche nach der Stecknadel im Heuhaufen gleichkommt.

Bevor Sie sich an das Herausfiltern der tatsächlichen Falschmeldungen machen, sollten Sie durch geeignete Scan-Einstellungen das Rauschen zumindest unterdrücken. Dabei helfen die folgenden Schritte:

- **Gemeinsame Lücken aller Hosts identifizieren**: Als Erstes sollten Sie sich die Mühe machen und all jene Lücken identifizieren, die allen zu scannenden Hosts gemein sind. Ein Beispiel für eine solche Lücke (wobei der Begriff Lücke den wahren Umstand nicht exakt beschreibt, sondern man besser von einem potenziellen Problem sprechen sollte) ist beispielsweise ein FTP-Port, der anonyme Verbindungen entgegennimmt. Auch ein offener Port 80 bei einem Proxy Server oder das Aktivieren von SNMP wird als mögliche Lücke erkannt. In einem solchen Fall wäre es

natürlich sinnvoll, wenn ein entsprechend „globaler“ Hinweis ausgegeben wird, der sich nicht auf alle Hosts einzeln bezieht.

- **Nicht zutreffende Anwendungen, Dienste und Umgebungen entfernen**: Die Informationsflut lässt sich durch eine weitere einfache Überlegung eindämmen. So könnten Sie beispielsweise die Traceroute-Informationen ausschalten, wenn man Hosts eines internen Netzes scannt. In diesem Fall spielen diese Informationen keine Rolle. Ähnlich ist es mit Scans, die rein informativen Charakter haben, beispielsweise die Identifikation des Betriebssystems oder von Ports.

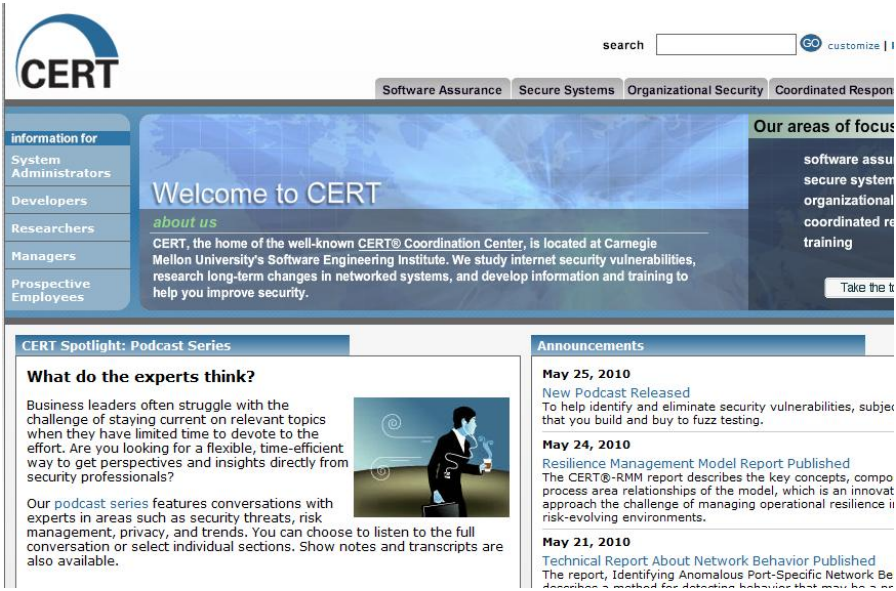
Durch das Deaktivieren solcher Informationen entledigen Sie sich bereits vieler, meist nicht benötigter Daten. Sollte man dennoch feststellen, dass hier und da zusätzlicher Informationsbedarf besteht, so kann man das Auslesen bestimmter Daten gezielt reaktivieren.

Nachdem Sie die Testumgebung eingeschränkt haben, geht es darum, die Informationen, die der Bericht ausgibt, der Reihe nach auf ihre Relevanz hin zu untersuchen. Dazu müssen Sie wohl oder übel Meldung für Meldung durchgehen, diese beurteilen und gegebenenfalls zusätzliche Informationen einholen. Möglicherweise muss ein Problem auch manuell verifiziert werden.

Bei der Analyse der Meldung ist es wichtig, dass Sie die Meldung im Kontext der jeweiligen Umgebung sehen. Dabei müssen Sie sich fragen, ob das jeweilige Problem tatsächlich ein kritisches ist oder nicht. Aber auch ein Verständnis des Problems ist erforderlich. Nur wenn man die Hintergründe kennt, kann man ein Problem richtig einschätzen. Hierfür greifen Sie am besten auf folgende Online-Quellen zurück:

- Securityfocus (*http://www.securityfocus.com*)
- Packet Storm (*http://www.packetstormsecurity.net*)
- CERT (*http://www.cert.org*)
- Open Source Vulnerability Database (*http://www.osvdb.org*)
- SecuriTeam (*http://www.securiteam.com*)

Auf diesen Websites finden Sie mehr als ausreichend Informationen zu allen relevanten Sicherheitslücken. Auch Lösungen und Workarounds gehören meist dazu.



Ohne aktuelle Hintergrundinformationen lässt sich die Relevanz einer Sicherheitslücke kaum abschätzen. Die notwendigen Informationen bieten CERT & Co.

Leider lassen sich solche Dinge nicht richtig lernen. Das Gespür und die richtige Einordnung für die Relevanz eines Problems kommen meist mit zunehmender Erfahrung im Umgang mit einem Werkzeug wie OpenVAS. Hilfreich sind natürlich fundierte Kenntnisse der Netzwerktechnik.

Als Nächstes sollten Sie bei kritischen Meldungen die Ergebnisse von OpenVAS manuell verifizieren. Dazu können Sie sich auch eines anderen Scanners bedienen. Verschiedene Probleme lassen sich beispielsweise auch durch Telnet-Sessions oder mithilfe von Netcat überprüfen. HTTP-spezifische Probleme kann man auch mit einem Browser verifizieren. Gibt OpenVAS merkwürdige Ergebnisse aus, so kann man sich im Zweifelsfall auch an den Entwickler des jeweiligen Skripts wenden. Auch die OpenVAS-Mailingliste ist ein guter „Ansprechpartner“.

Um dem Problem der Falschmeldungen beizukommen, gibt es verschiedene Ansätze. Einige sind oben bereits angeklungen. Am einfachsten ist es, man überlässt sie sich selbst. In kleinen Netzen mit einer Handvoll Rechnern mag das eine praktikable Lösung sein. Doch in der Praxis taugt sie dort nicht, wo Systeme eine kritische Aufgabe, wie beispielsweise die Bereitstellung von Applikationsdiensten, wie ein Content-Managementsystem, Mailserver oder Außenbindung, erfüllen. Der

bessere Weg ist sicherlich das Schließen kritischer Lücken durch das Einspielen von Patches, das Anpassen entsprechender Konfigurationseinstellungen etc.

Je nach Art der Meldungen können Sie diese auch gezielt durch das Deaktivieren von Plug-ins abstellen. Diese Vorgehensweise eignet sich allerdings nur für routinierte Anwender, die auch tatsächlich wissen, was sie machen. Schließlich besteht die Gefahr, dass einem eine echte Lücke durch die Lappen geht.

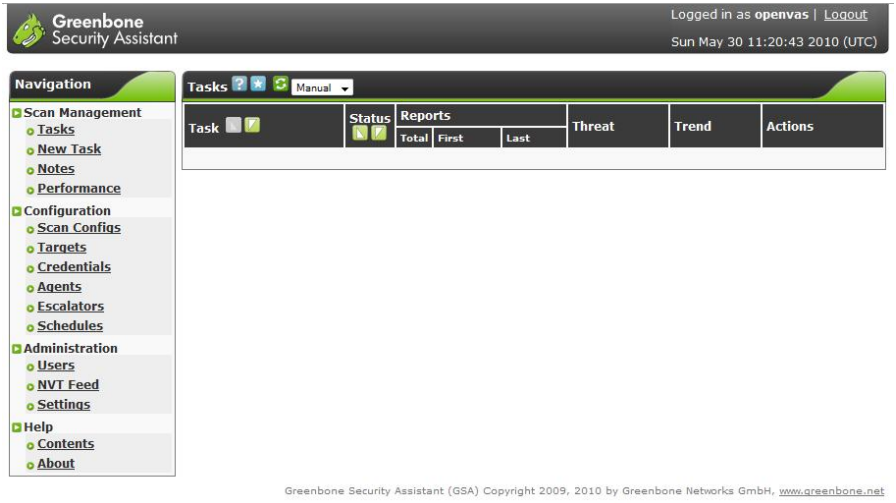
NASL-Profis können sogar noch einen Schritt weitergehen und Änderungen an einem Testskript vornehmen. Die meisten Plug-in-Entwickler erlauben Anpassungen an eigene Anforderungen. Im Zweifelsfall sollte man allerdings den Entwickler kontaktieren. Das gezielte Deaktivieren von Plug-ins ist aufgrund der übersichtlichen Schnittstellen der verfügbaren Clients einfach.

Nachdem Sie die Anpassung vorgenommen haben, sollten Sie einen Testdurchlauf starten und die geänderten Einstellungen verifizieren, damit nicht weitere Probleme oder Ungereimtheiten auftreten.

6 Die Zukunft des Scannens: GSA

Mit dem sogenannten Greenbone Security Assistant (GSA), dem OpenVAS-Administrator und dem OpenVAS-Manager hat das Entwickler-Team drei vollständig neue Komponenten für das OpenVAS-System entwickelt.

Mir persönlich hat es der GSA angetan. Dabei handelt es sich um ein webbasiertes Tool, über das Sie alle wichtigen Aufgaben bei der Durchführung Ihrer Sicherheitschecks durchführen können. Es handelt sich also um eine webbasierte Alternative zum OpenVAS-Client. In einigen Bereichen hat der GSA sogar mehr zu bieten also der Client.



Ein erster Blick auf den Greenbone Security Assistant.

6.1 Greenbone Security Assistant

Der Greenbone Security Assistant stellt Ihnen einen Webserver und damit eine webbasierte Schnittstelle zur Verfügung, über die alle wichtigen Aktionen für die Durchführung Ihrer Scans möglich sind. Der Webserver wurde mit libmicrohttpd (*http://www.gnu.org/software/libmicrohttpd/*) realisiert.

Sie können Ihre bestehenden Scan-Profile von einem Desktop-Client in den GSA importieren und dort nutzen. Der vielleicht wichtigste Vorteil: Selbst weniger technisch versiertes Personal oder das Management können pezielle Prüfungen durchführen.

Wie bei dem OpenVAS-Client können Sie mit wenigen Mausklicks PDF-, HTML- und XML-Exporte von den Berichten erstellen. Eine weitere Besonderheit: Der GSA zeigt auf einfache Weise den Sicherheitsstatus und dessen Trend an.

Zum besseren Verständnis: Der OpenVAS-Manager ist die neue OMP-Schicht. Sie sitzt grundsätzlich zwischen Scanner und Client. Damit wird es beispielsweise ermöglicht, Ergebnisse schon während eines laufenden Scans einzusehen. Der Vorteil: Sie müssen also nicht, wie beim Client, warten, dass der Scan fertig ist.

Der Administrator ist hingegen eine optionale Komponente. Damit können Sie Benutzer verwalten und beispielsweise auch den NVT Feed aktualisieren. Der GSA ist das GUI für beide.

Ein weiterer Vorteil: Sie müssen Benutzer nicht mehr an der Kommandozeile anlegen, sondern können das bequem aus der GUI heraus erledigen. Für den Einsatz spricht außerdem, dass Sie Scans zeitlich steuern können. Auch der automatische Versand einer Mail ist möglich, wenn ein Scan den Staus ändert.

Bevor wir uns der Installation und der Nutzung der drei Komponenten zuwenden, geben wir Ihnen zunächst einen Überblick, welche GSA-Funktionen welcher Komponente zuzuordnen sind.

GSA-Menü	Komponenten *	Aufgabe
Scan Management		
Tasks	M	Übersicht der Tasks (Scans)
New Task	M	Einen neuen Task (Scan) erstellen
Notes	M	Notizen zu Tasks bzw. zu den Ergebnissen eines Tasks
Performance	M	Hier werden noch externe Programme benötigt. Dort werden dann per rrdtool Grafiken zur Systemauslastung angezeigt.
Configuration		
Scan Configs	M	Welche Plugins sollen aktiv sein, welche Ports gescannt werden etc.?
Targets	M	Die Zielsysteme eines Tasks

Credentials	M	Für Local Security Scans (ssh oder smb)
Agents	M	Für SLAD/Winslad. Nicht wirklich wichtig derzeit.
Escalators	M	Benachrichtigungen (derzeit nur Mail)
Schedules	M	Scans planen und zu bestimmten Zeiten ausführen.
Administration		
Users	A	User verwalten
NVT Feed	A	Feed aktualisieren
Settings	A	Die Einstellungen des Scanners global ändern. Der Openvas-Administrator muss dazu mit der Option *--enable-modify-settings* gestartet werden.

* (M = Manager, A = Administrator)

6.2 GSA, Administrator und Manager installieren

Bevor Sie sich an die Installation von GSA, dem OpenVAS-Administrator und dem Manager machen, sollten Sie sicherstellen, dass die OpenVAS-Libraries installiert sind. Danach können Sie an die Installation der drei weiteren Komponenten gehen. Die Reihenfolge ist dabei prinzipiell gleich.

Am besten beginnen Sie mit dem Manager, gefolgt von dem Administrator und dem GSA. Beginnen wir mit der Installation des Managers. Bevor Sie damit loslegen, sollten Sie prüfen, dass auf dem System folgende Komponenten installiert sind.

- cmake
- glib-2.0
- gnutls
- libuuid (from e2fsprogs)
- openvas-libraries 3.0.3 oder höher
- sqlite3

Für die Dokumentation sind außerdem folgende Komponenten relevant:

- Doxygen
- Xmltoman
- sqlfairy

Tipp:

Auf der beiliegenden CD finden Sie eine Live-Installation, die Ihnen das direkte Arbeiten mit der OpenVAs-Umgebung erlaubt. Nach dem Hochfahren ist eine vollständige Umgebung samt OpenVAS-Server, -Client und auch GSA verfügbar. Für weitere Details zum Einsatz sei auf Anhang D verwiesen.

Sollte */openvas/bin* nicht im PATH enthalten sein, fügen Sie dies hinzu, weil *libopenvas-config* benötigt wird. Das geschieht mit folgendem Kommando:

```
$ export PATH=$PATH:/pfad/zu/ihrer/installation/bin/
```

Wenn Sie eine Standardinstallation besitzen, fügen Sie als Nächstes folgenden Befehl aus:

```
$ cmake
```

Anschließend sollten Sie folgende Kommandos ausführen:

```
$ make
$ make doc
$ make doc-full
$ make install
$ make rebuild_cache
```

Beachten Sie, dass der OpenVAS-Manager sich als Benutzer *om* mit dem Scanner verbindet und dabei die zertifikatsbasierte Zugriffsvariante verwendet. Daher müssen Sie einen entsprechenden Benutzer samt Zertifikat anlegen. Dazu verwenden Sie am besten das Skript *openvas-mkcert-client* des OpenVAS-Servers.

Für das Erstellen eines Benutzers *om* samt Zertifikat und der anschließenden Registrierung des Benutzers beim OpenVAS-Server genügt auch eine Befehlszeile:

```
$ openvas-mkcert-client -n om
```

Beachten Sie außerdem, dass verschiedene Funktionen des OpenVAS-Managers auf zusätzliche Module zurückgreifen. Für das Erzeugen von

- PDF-Berichten benötigen Sie pdflatex
- HTML-Berichten benötigen Sie xmltproc
- Berechtigungs-RPM-/DEB-Paketen benötigen Sie
    - RPM
    - Fakeroot
    - alien

Wenn der OpenVAS-Daemon auf Port 9391 ausgeführt wird, so starten Sie den OpenVAS-Manager mit folgendem Kommando auf Port 9390:

```
openvasmd --port 9390 --sport 9391
```

Der Client kann dann eine Verbindung mit Port 9390 herstellen.

Mit den beiden folgenden Kommandos führen Sie ein Update des Manager-NVT-Cache durch:

```
openvasmd --sport 9391 --update
openvasmd --sport 9391 --rebuild
```

Beachten Sie, dass der Cache jedes Mal aktualisiert werden muss, wenn der Scanner einen Abgleich mit dem NVT Feed durchführt.

Mit folgendem Befehl rufen Sie die weiteren Befehle des Managers ab:

```
openvasmd -h
```

Als Nächstes installieren Sie den Adminstrator. Zur Kompilierung führen Sie folgenden Befehl aus:

```
$ cmake .
```

Führen Sie dann *make* und *make install* aus.

Den Administrator starten Sie mit folgendem Befehl:

```
$ openvasad
```

Wenn Sie mehr über seine Nutzung wissen wollen, verwenden Sie folgenden Befehl:

```
$ openvasad --help
```

Damit sind der Manager und der Administrator einsatzbereit und Sie müssen sich nur noch der Installation des GSA widmen.

Die Systemvoraussetzungen für die Installation des Greenbone Security Assistant sind mit den oben genannten weitgehend identisch. Daher sind hier keine weiteren Arbeiten mehr erforderlich. Bei einer OpenVAS-Standardinstallation führen Sie zunächst folgenden Befehl aus:

```
$ cmake .
```

Anschließend die folgenden:

```
$ make
$ make doc
$ make install
$ make rebuild_cache
```

Das OMP verwendet eine zertifikatbasierte Anmeldung, um ein Mehr an Sicherheit zu bieten. Dazu ist die Zertifikatgenerierung erforderlich, da sich der OpenVAS-Manager immer mit dem Scanner als User *om* verbindet und ausschließlich die zertifikatbasierte Authentifizierung nutzt.

Sie müssen daher diesen Benutzer anlegen, das Zertifikat und den Schlüssel für den Benutzer generieren und diese dem Manager zur Verfügung stellen. Am einfachsten greifen Sie dabei zum Skript *openvas-mkcert-client* des OpenVAS-Scanners. Führen Sie dazu folgenden Befehl aus:

```
openvas-mkcert-client -n om -i
```

Damit der Manager das Zertifikat und den Schlüssel nutzen kann, müssen Sie diese in ein spezifisches Verzeichnis kopieren. Dazu führen Sie folgende Kommandos aus:

```
cp key_om.pem /var/lib/openvas/private/CA/clientkey.pem
cp cert_om.pem /var/lib/openvas/CA/clientcert.pem
```

Wird *openvasmd* (der OpenVAS Manager-Daemon) auf Port 9390 ausgeführt, so starten Sie den Greenbone Security Assistant-Daemon auf Port 443 mit folgendem Befehl:

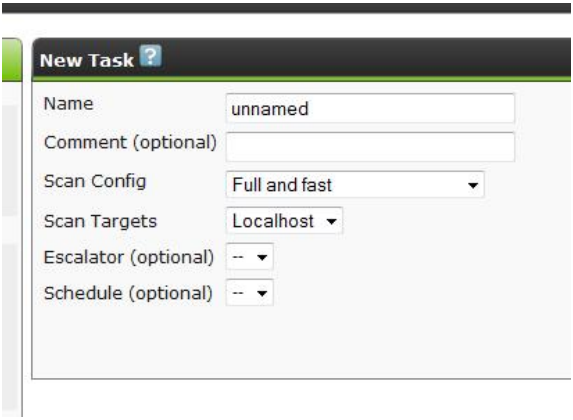
```
gsad --mport 9391
```

Alle verfügbaren Optionen rufen Sie mit der *Help*-Option ab:

```
gsad --help
```

Damit ist Ihre Administrator-Manager-GSA-Umgebung einsatzbereit und Sie können sich mit der Benutzerschutzschnittstelle vertraut machen.

Die Benutzerschnittstelle des GSA ist sehr übersichtlich strukturiert. In der Kopfzeile wird angezeigt, welcher Benutzer aktuell eingeloggt ist. Außerdem finden Sie hier den Logout-Link und das aktuelle Datum. Der Bereich unterhalb des Headers ist zweigeteilt: links die Navigationsleiste, rechts die dazugehörenden Funktionen. In der Fußzeile finden Sie die Copyright-Info der GSA-Entwickler.



Das Erstellen einer ersten Aufgabe.

6.3 Scan-Management

Die Funktionen des Bereichs *Scan-Management* dienen in erster Linie dem Erstellen und der Verwaltung von Scan-Aufträgen. Beim Zugriff auf den GSA präsentiert Ihnen das Tool standardmäßig die Task-Übersicht. Zu jeder Aufgabe werden die Bezeichnung, der Status, die Berichte, die Threads, die Trends und verfügbare Aktionen aufgeführt.

Um eine erste Aufgabe anzulegen, folgen Sie dem Link *New Task* und spezifizieren in dem zugehörigen Dialog die Eigenschaften:

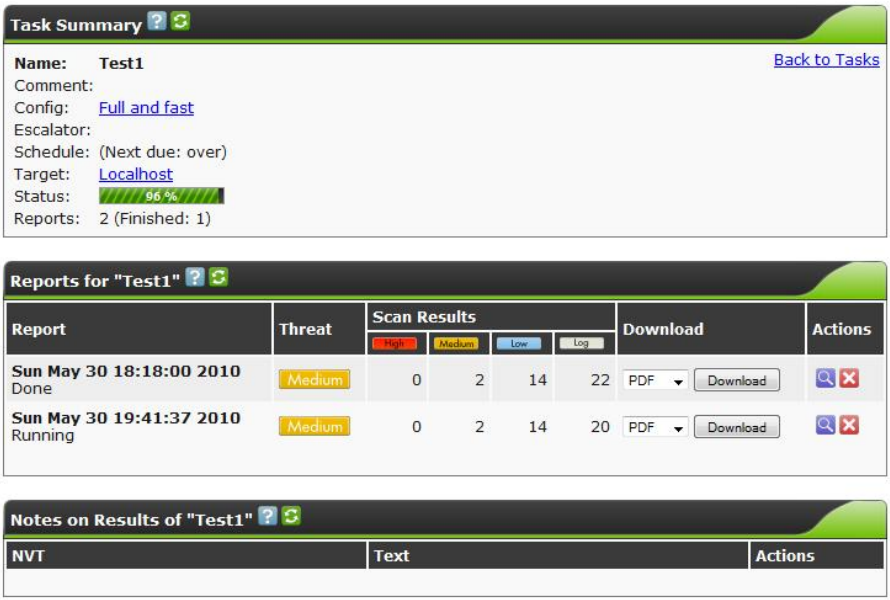
- **Name**: Die Bezeichnung der Aufgabe.
- **Comment**: Hier hinterlegen Sie einen optionalen Kommentar.
- **Scan Config**: Mit diesem Auswahlmenü wählen Sie gewünschte Scan-Konfiguration aus. Sie haben die Wahl zwischen verschiedenen vorbereiteten Varianten, die Sie unter *Configuration> Scan Configs* einsehen, bearbeiten und neu anlegen können:
    - Full and fast
    - Full and fast ultimate
    - Full and very deep
    - Full and very deep ultimate
    - IT-Grundschutz Scan
    - empty
- **Scan Targets**: Dieses Auswahlmenü erlaubt Ihnen die Auswahl der zu untersuchenden Ziele. Dabei stehen Ihnen die unter *Configuration> Targets* angelegten Ziele zur Auswahl.
- **Escalator**: Optional kann ein Trigger ausgewählt werden, der bei bestimmten Ereignissen ausgelöst wird. Auch diese müssen erst in der Konfiguration angelegt werden.
- **Schedule**: Dieses Auswahlmenü erlaubt die Wahl von Zeitplänen für die zeitliche Steuerung des Scan-Vorgangs. Sie müssen ebenfalls zuerst angelegt werden.

Um die erste Aufgabe zu speichern, klicken Sie auf die Schaltfläche *Create task*. Sie landen automatisch in der Task-Übersicht. Dort zeigt Ihnen die Tabelle an, ob

es sich um einen neuen oder bereits ausgeführten Scan handelt. Bei Letzterem finden Sie in der Spalte *Reports* die Berichtanzahl und können über den Anzahl bzw. Datum-Link auf die Berichtergebnisse zugreifen.

In der Spalte *Threat* präsentiert Ihnen die Task-Übersicht die Gefahreneinschätzung. Auch hier kommen Farben zur Kennzeichnung der Bewertung zum Einsatz. Über die *Actions*-Spalte können Sie die Ausgabe starten, anhalten, beenden, die Details abrufen und die Aufgabe editieren.

Wenn Sie eine erste Aufgabe angelegt und dann ausgeführt haben, können Sie aus der Task-Übersicht heraus mit einem Klick auf das Lupen-Symbol (Details) eine Zusammenfassung des Testdurchlaufs abrufen.



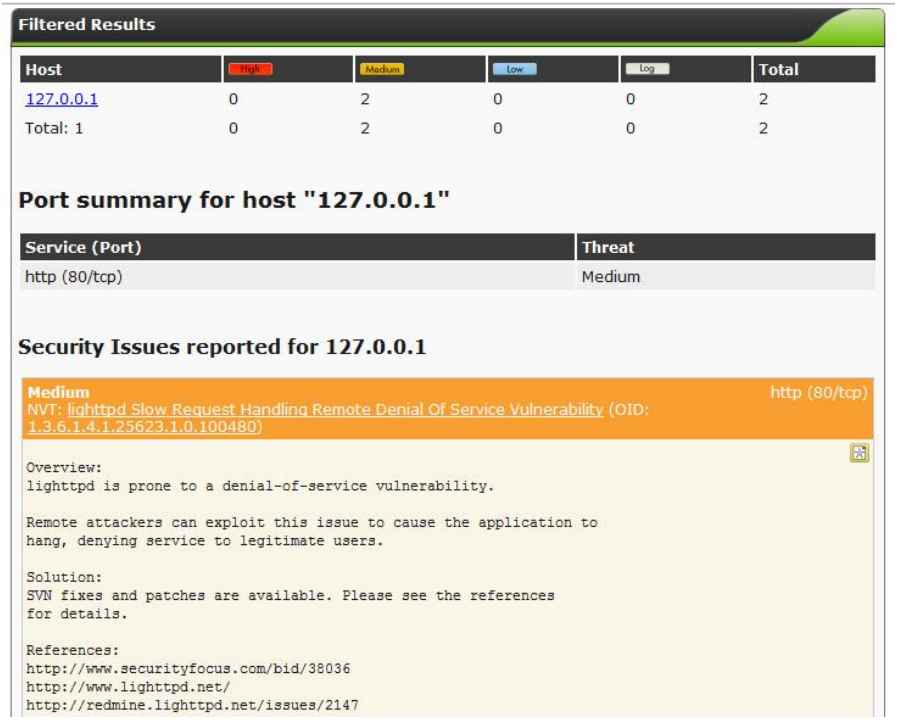
Die Ausgabenzusammenfassungen des ersten Tests.

Die Testzusammenfassung präsentiert Ihnen insbesondere im Bereich *Scan Results* die Ergebnisse und die Risikoeinteilung. Die Spalte *Download* erlaubt den Export des Reports in verschiedene Exportformate.

Wenn Sie sich für weitere Details eines Berichts interessieren, klicken Sie in der Task-Zusammenfassung erneut auf das Lupensymbol. Auf der zugehörigen Seite

finden Sie zunächst eine Zusammenfassung der wichtigsten Berichtdaten. Es folgt der Bereich *Result Filtering*, über den Sie die in dem Bericht enthaltenen Informationen beispielsweise auf bestimmte Risikostufen beschränken können.

Hier schließt sich der Bereich *Filtered Results* an, der Ihnen eine Vielzahl an weiteren Detailinformationen zu bieten hat. Zu allen schweren und mittelschweren Risiken werden außerdem Detailinformationen der NVT ausgegeben.



Die Berichtdetails.

Ein wesentliches Element eines anspruchsvollen Schwachstellenmanagements ist das Erstellen und Verwalten von Notizen. So können Sie die Detailinformationen um wichtige Zusatzinformationen ergänzen.

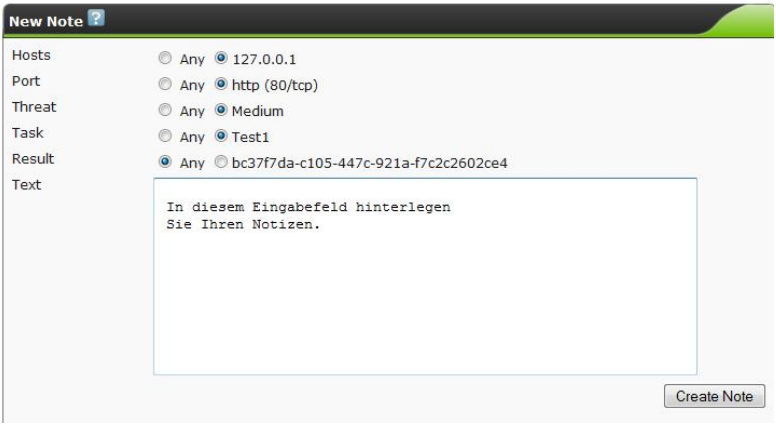
Der GSA stellt Ihnen eine leistungsfähige Notizfunktion zur Verfügung. Das Besondere daran: Ihre Anmerkungen können auch in die Berichtexporte aufgenommen werden.

Das Anlegen ist einfach: Klicken Sie in den Ergebnissen auf das Icon *Add Note* und bestimmen Sie im *New Note*-Dialog die Eigenschaften der Notiz.

Das Hinzufügen einer Notiz.

Für das Hinzufügen von Notizen steht Ihnen das Eingabefeld *Text* zur Verfügung. Mit einem Klick auf die Schaltfläche *Create Note* legen Sie die Anmerkung an. Diese wird am Ende des Bereichs angehängt.

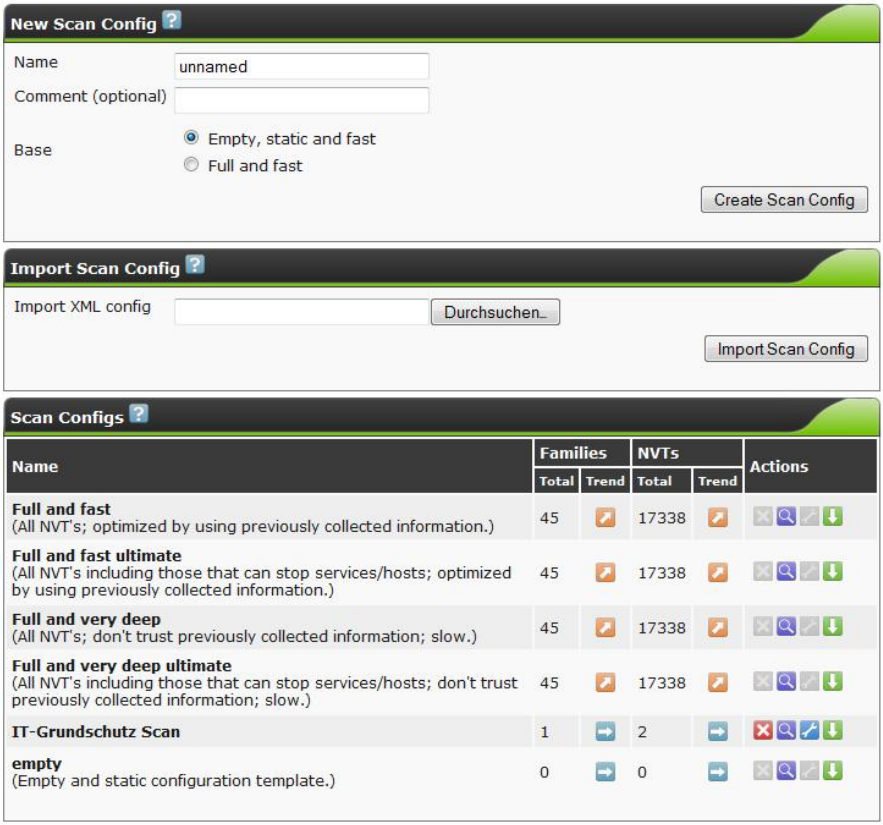
Nach dem Hinzufügen der Notiz finden Sie diese in der Notes-Verwaltung. Dort wird Sie tabellarisch samt NVT und einem Auszug aufgeführt. Auch das Bearbeiten ist über die Actions-Spalte möglich.



Eine erste Notiz entsteht.

6.4 Scan-Konfiguration

Der Bereich *Configuration* dient der Konfiguration der Scans. Hier bestimmen Sie, welche NVTs ausgeführt, welche Ziele ins Visier genommen und welche Zugangsdaten für lokale Tests verwendet werden. Außerdem können hier Agents (Drittanwendungen) sowie Warnungskriterien definiert und zeitliche Steuerungen angelegt werden.



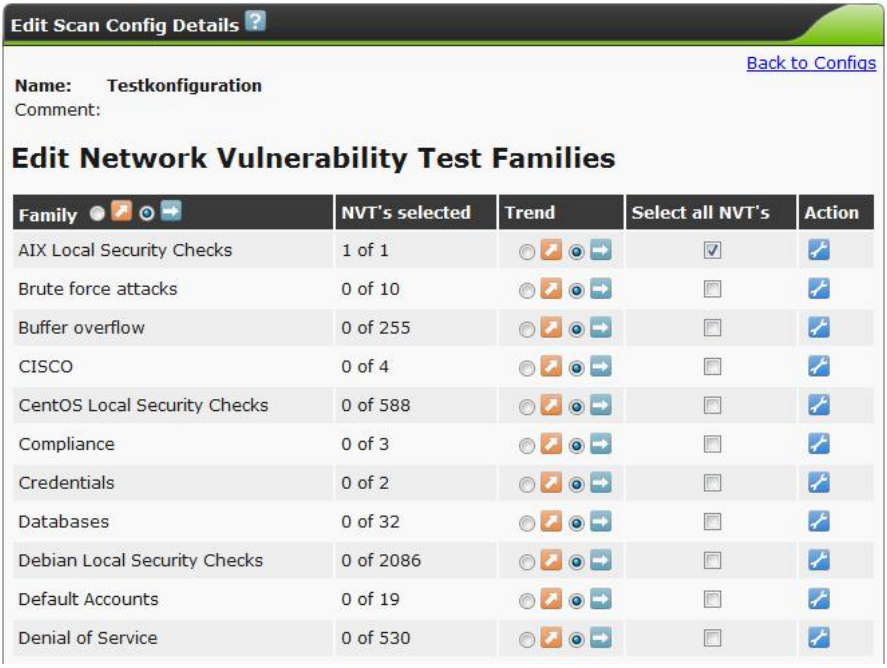
Die Übersicht der Scan-Konfiguration.

Wenn Sie im Bereich *Configuration* dem Link *Scan Configs* folgen, landen Sie in einem umfangreichen Formular, das Ihnen das Erstellen neuer Scan-Konfigurationen erlaubt. Auch der Import von bestehenden Konfigurationen ist möglich, solange diese XML-basiert sind.

Es folgt die Übersicht der eigentlichen Scan-Konfigurationen. In diesen können Sie nach Belieben und Anforderungen Testskripts zusammenfassen. Der GSA kommt bei einer Standardinstallation mit sechs vordefinierten Scan-Konfigurationen daher, deren Bezeichnungen oben schon gefallen sind. Die ersten vier sind identisch konfiguriert, können aber nicht editiert und auch nicht gelöscht werden. Auch die leere Konfiguration kann nicht gelöscht werden.

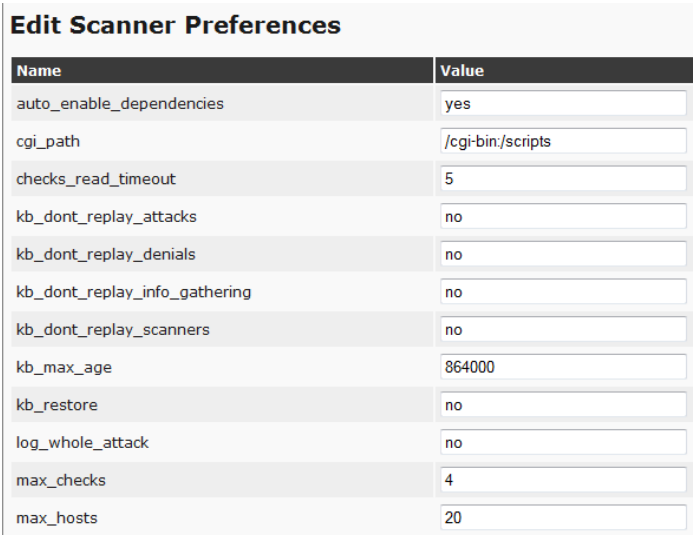
Auf die Konfiguration *IT-Grundschutz* können Sie hingegen alle Aktionen anwenden, diese also löschen, Details einsehen, sie bearbeiten und sie nach XML exportieren.

Interessante Zusatzinformationen liefern Ihnen übrigens die Spalten *Families* und *NVTs*. Hier werden Ihnen die Anzahl der verwendeten Scan-Familien und Skripts sowie die Trends angezeigt.



Das Editieren einer ersten eigenen Scan-Konfiguration.

Die Vorgehensweise beim Erstellen und Einrichten einer Scan-Konfiguration ist einfach: Erstellen Sie zunächst über den Bereich *New Scan Config* einen neuen Eintrag. Dann editieren Sie diesen in der Konfigurationsübersicht. Sie landen im Dialog *Edit Scan Config Details*. Der führt die verfügbaren Familien auf und erlaubt über die Action-Spalte auch die Auswahl von einzelnen Skripts. Unterhalb der Familienliste finden Sie die Scanner-Einstellungen. Auch diese Einstellungen können Sie entsprechend Ihren Vorstellungen bearbeiten. Weiter unten folgen die Skript-Einstellungen – soweit verfügbar.

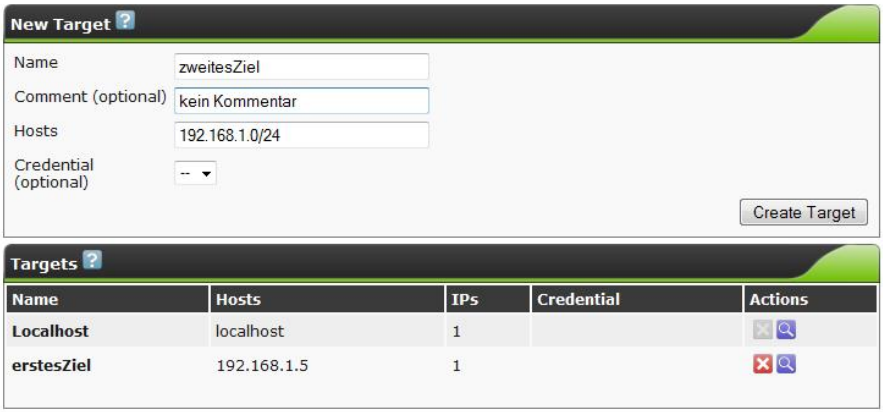


Das Bearbeiten der Scanner-Einstellungen.

Wenn Sie eine Scan-Konfiguration einer Aufgabe zugewiesen haben, dann wird auch das am Ende der umfangreichen Formularseite angezeigt. Auf die Details der unzähligen Einstellungen muss an dieser Stelle nicht mehr eingegangen werden, denn die sind in Kapitel 4 detailliert beschrieben.

Als Nächstes erlaubt Ihnen der *Configuration*-Bereich das Anlegen und Verwalten der Ziele. Auch das ist einfach. Um ein neues Ziel anzulegen, folgen Sie unter *Configuration* dem *Targets*-Link.

Hier weisen Sie der Zielkonfiguration eine Bezeichnung, optional eine Beschreibung und die Ziele selbst zu. Optional können Sie auch bereits angelegte Credentials für die Ausführung lokaler Scans nutzen. Mit einem Klick auf die Schaltfläche *Create Target* legen Sie die Zielkonfiguration an.



Das Anlegen einer neuen Zielkonfiguration.

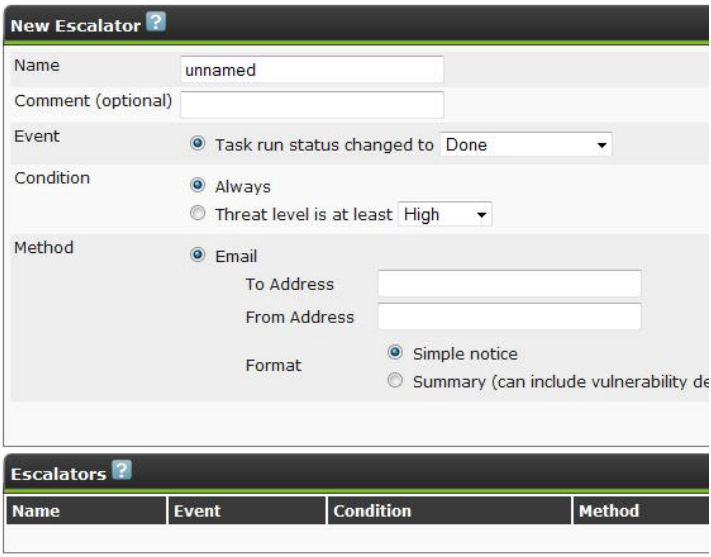
Nach dem Anlegen zeigt Ihnen der GSA – wie bei fast allen durchgeführten Aktionen – die ausgeführten Kommandos im Kopfbereich und unterhalb die erweiterte Zieltabelle an.

Ein Bearbeiten eines Zieleintrags ist bislang übrigens über die Actions-Spalte nicht möglich. Sie können Ihre Ziele lediglich löschen und einsehen.

Es folgen die Einstellungen des Bereichs *Credentials*. Wie Sie bereits wissen, sind diese Einstellungen für lokale Tests erforderlich, bei denen Sie sich in ein System oder einen Dienst einloggen müssen. Die Handhabung ist wieder einfach: Weisen Sie dem Eintrag eine Bezeichnung sowie das Log-in und das Passwort zu. Mit einem Klick auf *Create Credential* sichern Sie den Eintrag. Er kann dann beim Anlegen bzw. Bearbeiten einer neuen bzw. einer bestehenden Aufgabe verwendet werden. Die *Credentials*-Übersicht erlaubt ebenfalls nur das Löschen und Einsehen eines Eintrags.

Das *Agent*-Menü ermöglicht Ihnen das Einbinden von Drittprogrammen inklusive Installations- und Anwendungsdokumentationen.

Hinter dem Begriff *Escalators* verbirgt sich die Konfiguration für Hinweismeldungen, die beim Eintreten definierbarer Ereignisse ausgegeben werden. Diese Meldungen können den Aufgaben zugewiesen werden. Die Nutzung ist ebenfalls sehr einfach, denn das zugehörige Formular erlaubt das Anlegen und das Verwalten der Einstellungen.



Das Anlegen einer Hinweismeldung.

Um einen ersten *Escalator*-Eintrag zu erstellen, weisen Sie diesem eine Bezeichnung und optional wieder einen Kommentar zu. Im Auswahlmenü Event bestimmen Sie, welche Statusänderung die Ausgabe bewirkt. Sie haben die Wahl zwischen folgenden Einträgen:

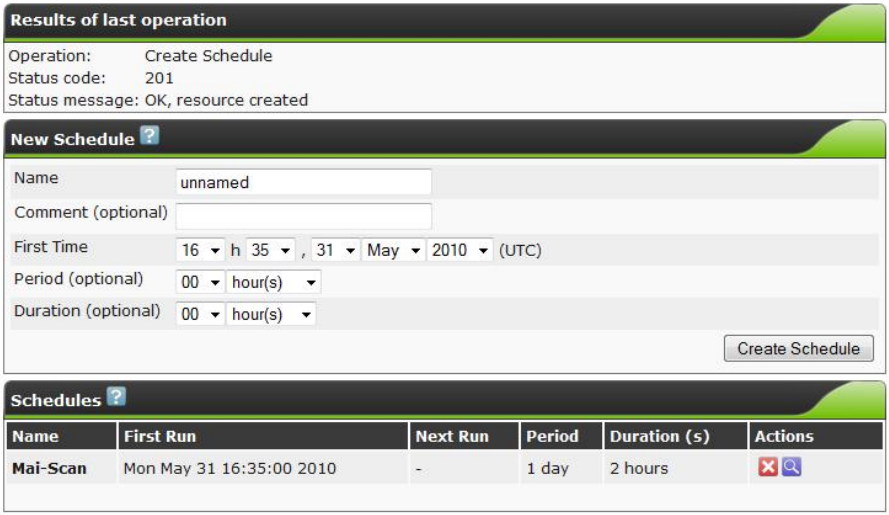
- Delete Requested
- Done
- New
- Requested
- Running
- Stop Requested
- Stopped

Als Kondition können Sie sich zwischen zwei Optionen entscheiden:

- **Always**: Die Warnung wird immer ausgegeben

- **Threat level is at least**: Die Bewertungsstufe muss mindestens den über das Auswahlmenü zu bestimmenden Wert besitzen. Mögliche Werte sind High, Medium, Low und Log.

Im Bereich *Method* bietet der GSA bislang nur die Möglichkeit, die Warnung per E-Mail an eine bestimmte E-Mail-Adresse mit der im Eingabefeld *From Address* anzugebenden Absenderadresse zu verschicken. Beim E-Mail-Versand können Sie unter *Format* außerdem festlegen, ob lediglich eine einfache Notiz oder eine Zusammenfassung (womöglich mit Details zu den Verwundbarkeiten) versandt wird. In Zukunft sind vielleicht auch SMS-Meldungen möglich. Die Konfiguration legen Sie mit einem Klick auf die Schaltfläche *Create Escalator* an.



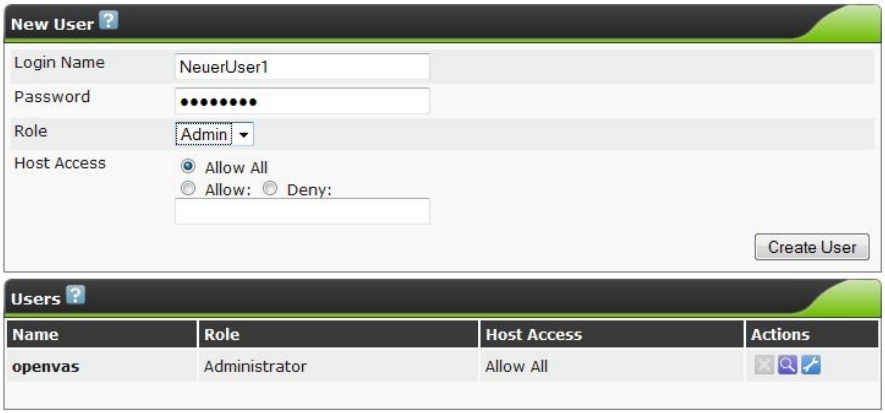
Der Zeitplaner erlaubt die zeitlich gesteuerte Ausführung von Sicherheits-Scans.

Ein echtes Highlight des GSA ist der Scheduler. Er erlaubt die zeitlich gesteuerte Ausführung Ihrer Scans. Sie legen einfach durch Angabe einer möglichst aussagekräftigen Bezeichnung den Zeitpunkt der ersten Ausführung, der Wiederholung und die Dauer die Eigenschaften fest. Diese Steuerung können Sie dann in der Aufgabenverwaltung Ihren Einträgen zuweisen.

In der Aufgabenverwaltung erkennen Sie in der *Actions*-Spalte direkt, welche Aufträge zeitlich gesteuert und welche manuell ausgeführt werden.

6.5 OpenVAS-Konfiguration

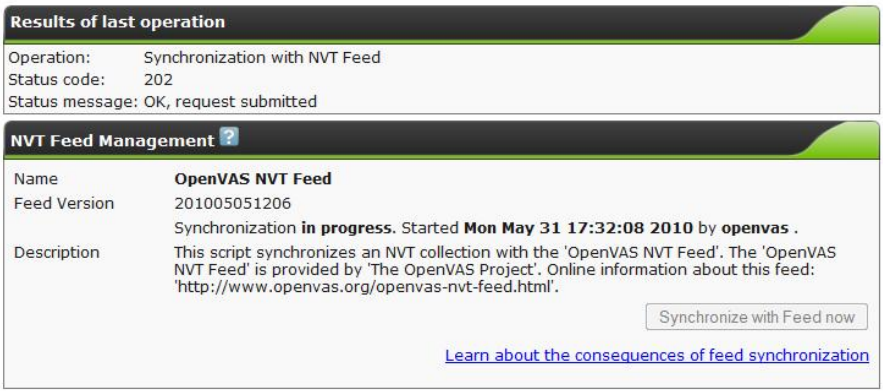
Mit dem Bereich *Configuration* folgen die Einstellungen, für die der OpenVAS-Administrator zuständig ist: die Benutzerverwaltung, die NVT-Aktualisierung und die Scanner-Einstellungen.



Das Anlegen eines neuen OpenVAS-Benutzers.

Das Highlight ist sicherlich die Benutzerverwaltung, mit der das Anlegen und Bearbeiten von neuen bzw. bestehenden Benutzern einfach wird. Um neben dem bereits erzeugten OpenVAS weitere Benutzer einzuführen, folgen Sie in der Navigationsleiste dem Link *Users*. Weisen Sie im Bereich *New User* dem Benutzer einen Log-in-Namen und ein Passwort zu. Der Log-in-Name darf maximal 80 Zeichen, das Passwort höchstens 40 Zeichen lang sein.

Im Auswahlmenü *Role* bestimmen Sie die Rolle. Sie haben die Wahl zwischen den Optionen *User* und *Admin*. Bestimmen Sie als Nächstes den Host-Zugriff. Mit einem Klick auf *Create User* legen Sie den ersten neuen Benutzer an. Sie landen automatisch in der Benutzerübersicht und können dort die Benutzer bearbeiten, löschen und weitere anlegen.



Die Aktualisierung der NVTs läuft.

Hinter dem Administration-Link *NVT-Feed* verbirgt sich eine einfache Funktion: Sie können mit einem Klick auf die Schaltfläche *Synchronize with Feed now* Ihre lokale Skripts-Sammlung auf den neuesten Stand bringen. Dafür sind allerdings Admin-Berechtigungen erforderlich.

Beachten Sie, dass dieser Vorgang beim Einsatz der beiliegenden Live-CD nur begrenzt Sinn macht.

Wenn Sie wissen wollen, mit welchen Einstellung der OpenVAS-Server ausgeführt wird, so können Sie diese über den GSA bequem unter *Settings* einsehen. Hier präsentiert Ihnen die Webschnittstelle die Einstellung aus der Open-VAS-Server-

Konfigurationsdatei */etc/openvas/openvassd.conf*. Änderungen sind hier allerdings nicht möglich.

Scanner Settings

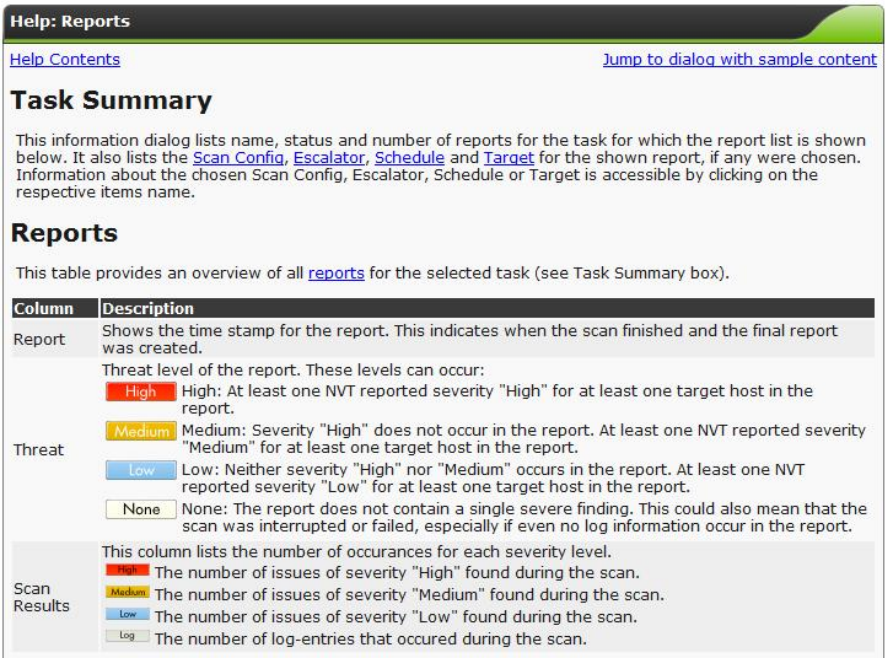
From file: /etc/openvas/openvassd.conf

Setting	Value
plugins_folder	/usr/lib/openvas/plugins
cache_folder	/var/cache/openvas
include_folders	/usr/lib/openvas/plugins
max_hosts	2
max_checks	4
be_nice	no
logfile	/var/log/openvas/openvassd.messages
log_whole_attack	no
log_plugins_name_at_load	no
dumpfile	/var/log/openvas/openvassd.dump
rules	/etc/openvas/openvassd.rules
cgi_path	/cgi-bin:/scripts
port_range	default
optimize_test	yes
checks_read_timeout	5
non_simult_ports	139, 445
plugins_timeout	320

Die Konfiguration des OpenVAS-Servers.

In den vorangegangenen Abschnitten haben Sie den GSA und seine wichtigsten Funktionen und Möglichkeiten kennengelernt. Wenn Sie weitere Informationen zu

dieser wirklich sehr gelungenen Schnittstelle benötigen, sollten Sie einen Blick in die integrierte Hilfe werfen. Dort finden Sie weitere Details.



Help: Reports

Help Contents | Jump to dialog with sample content

Task Summary

This information dialog lists name, status and number of reports for the task for which the report list is shown below. It also lists the Scan Config, Escalator, Schedule and Target for the shown report, if any were chosen. Information about the chosen Scan Config, Escalator, Schedule or Target is accessible by clicking on the respective items name.

Reports

This table provides an overview of all reports for the selected task (see Task Summary box).

Column	Description
Report	Shows the time stamp for the report. This indicates when the scan finished and the final report was created.
Threat	Threat level of the report. These levels can occur: High: At least one NVT reported severity "High" for at least one target host in the report. Medium: Severity "High" does not occur in the report. At least one NVT reported severity "Medium" for at least one target host in the report. Low: Neither severity "High" nor "Medium" occurs in the report. At least one NVT reported severity "Low" for at least one target host in the report. None: The report does not contain a single severe finding. This could also mean that the scan was interrupted or failed, especially if even no log information occur in the report.
Scan Results	This column lists the number of occurances for each severity level. The number of issues of severity "High" found during the scan. The number of issues of severity "Medium" found during the scan. The number of issues of severity "Low" found during the scan. The number of log-entries that occured during the scan.

Vorbildlich: Die Hilfe des GSA erläutert auch alle farblichen Aspekte der Benutzerführung.

7 OpenVAS für Fortgeschrittene

Wenn Sie OpenVAS optimal einsetzen wollen, so ist ein grundsätzliches Verständnis der Architektur und der internen Abläufe nützlich. Mit diesem Hintergrundwissen sind Sie bestens gerüstet, um Berichte besser zu verstehen. Auch auftretende Probleme können Sie einfacher lösen.

Wie bei vielen anderen vergleichbaren Werkzeugen ist der Vorgang beim Erkennen von Verwundbarkeiten in mehrere Abschnitte unterteilt. Dabei sind die einzelnen Etappen abhängig von der erfolgreichen Ausführung der vorangegangenen Schritte. Die einzelnen Schritte sind ihrerseits durch die Plug-ins, also die Testskripts, gekennzeichnet. Jedes Plug-in ist Teil eines solchen Schrittes, wobei es die bereits erwähnten Abhängigkeiten gibt, dass Skripte Ergebnisse anderer Tests benötigen, um selbst ausgeführt zu werden.

Wichtig ist in diesem Zusammenhang, dass es auch Plug-ins gibt, die bei bestimmten Testschritten nicht deaktiviert werden können, da andernfalls kein sinnvolles Testen und Berichten möglich ist.

7.1 Interne Abläufe

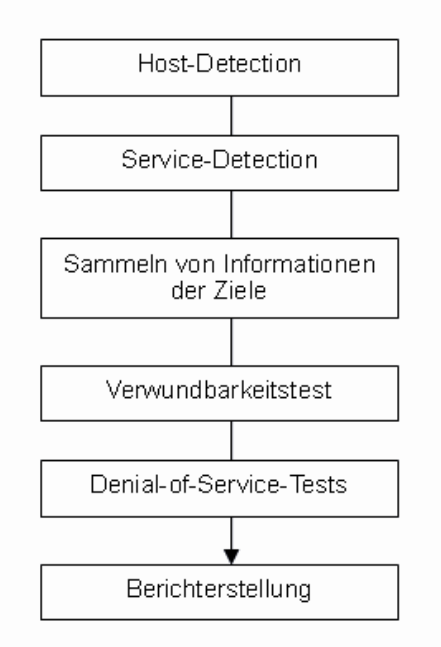
Vor den eigentlichen Scan-Vorgängen führt OpenVAS zwei vorbereitende, aber dennoch sehr wichtige Schritte aus: Zunächst wird eine sogenannte Host-Detection durchgeführt, um festzustellen, ob das gewünschte System überhaupt verfügbar ist. Als Nächstes wird eine Service-Detection durchgeführt. Dabei geht es darum, festzustellen, welche Ports auf dem Ziel verfügbar sind und welche Dienste hinter diesen Schnittstellen laufen. Die erfolgreiche Ausführung dieser beiden Schritte ist Grundvoraussetzung für die weitere Ausführung von OpenVAS.

Im nächsten Schritt macht sich OpenVAS an das Sammeln von Informationen. Dabei werden Daten der einzelnen Anwendungen und Dienste zusammengetragen, insbesondere der Name und die Version der jeweiligen Anwendung. Anhand dieser Informationen kann OpenVAS bestimmen, welche Verwundbarkeiten für die verfügbaren Dienste zutreffen können. Dabei wird ein Abgleich mit den Plug-in-Informationen durchgeführt.

Der Scanner spezifiziert außerdem, welche Produkte, Service Packs, Hotfixes, Patches etc. auf dem Ziel installiert sind. Diese Informationen sind für die Plug-ins wichtig, da auf dieser Grundlage entschieden wird, welche Tests durchgeführt

werden und welche nicht. Die gewonnenen Informationen sind also für die Testdurchführung essenziell.

Die beiden folgenden Schritte führen Verwundbarkeits- und Denial-of-Service-Tests durch. In dieser Phase wird auf Grundlage der zuvor ermittelten Informationen eine Vielzahl von Plug-ins geladen und ausgeführt. Welche Plug-ins das sind, hängt in erster Linie von den Diensten und Anwendungen wie auch von den Benutzervorgaben ab. Den Abschluss bildet die Berichterstellung, in der die gesammelten Ergebnisse zusammengefasst und in die verschiedenen Formate exportiert werden.



Ablauf eines typischen Scan-Vorgangs.

Wenn OpenVAS mehrere IP-Adressen scannt, so werden Dutzende, im Extremfall sogar Hunderte von Subprozessen erzeugt. Außerdem wird für jede Client-Verbindung ein neuer Prozess erzeugt, der die Kommunikation zwischen Client und Server steuert. Als C-basiertes Programm ist OpenVAS außerordentlich schnell in der Verarbeitung der Daten. Auch die Skripts waren ursprünglich in C programmiert, Letztere wurde aber inzwischen durch die C-nahe Sprache NASL abgelöst.

OpenVAS basierte auf einer typischen Client-Server-Architektur. Der Client steuert den Server, der für die eigentliche Durchführung der Tests und das Speichern sowie das Verwalten der gewonnenen Daten zuständig ist. Hat der Server die Tests beendet, so sind die Ergebnisse über den Client abrufbar. Die vorangegangene Abbildung zeigt, wie die verschiedenen Komponenten der Architektur interagieren.

Nessus lenkt die Plug-ins bezüglich dreier Regeln. Die erste Regel sorgt dafür, dass kein Portscanner geladen wird, der womöglich das Ziel zum Absturz bringt. Die zweite Regel ist für die Abhängigkeiten der Plug-ins untereinander zuständig. Sie stellt sicher, dass die Plug-ins in der adäquaten Reihenfolge gestartet werden. Schließlich sorgt die letzte Regel dafür, dass die globalen Einstellungen geladen werden.

Schauen wir uns an, was bei der Host-Detection passiert. Wichtigste Aufgabe dieses Schritts ist es, festzustellen, ob das Ziel verfügbar ist oder nicht. Dazu greift Nessus auf Ping zurück und setzt das Skript *ping_host.nasl* ein. Dieses Skript kann die Verfügbarkeit von Hosts auch dann erkennen, wenn diese beispielsweise ICMP-Echo-Requests nicht beantworten.

Alle IP-Adressen, die nicht ansprechbar sind, werden im Report unter *general/tcp* in den Log-Messages als nicht erreichbar gekennzeichnet. Versucht man, einen einzelnen Host anzusprechen, der nicht auf Ping reagiert, so wird in der Regel ein leerer Bericht ausgegeben. Dieser Schritt ist in der Regel sehr zeitaufwendig. Daher sollte man den IP-Bereich nicht zu weit fassen, da ansonsten Scans auch schon mal mehrere Stunden dauern können.

Nachdem die erreichbaren Hosts als solche identifiziert sind, kommt die Diensterkennung zum Zug, die herauszulesen versucht, welche Dienste auf den jeweiligen Hosts verfügbar sind. Dazu setzt OpenVAS das Plug-in *ACT_SCANNER* ein. Für die eigentlichen Scan-Vorgänge greift OpenVAS auf verschiedene Tools zurück.

Dazu müssen die Tools zuerst in den Scan-Optionen aktiviert werden. Die von dem Scanner gesammelten Informationen werden dann in der Knowledgebase gespeichert. Liegt die gesamte Liste der Port-Informationen vor, so startet OpenVAS sein Diensterkennungs-Plug-in, das sich am Auslesen der Diensttypen hinter jedem offenen Port versucht.

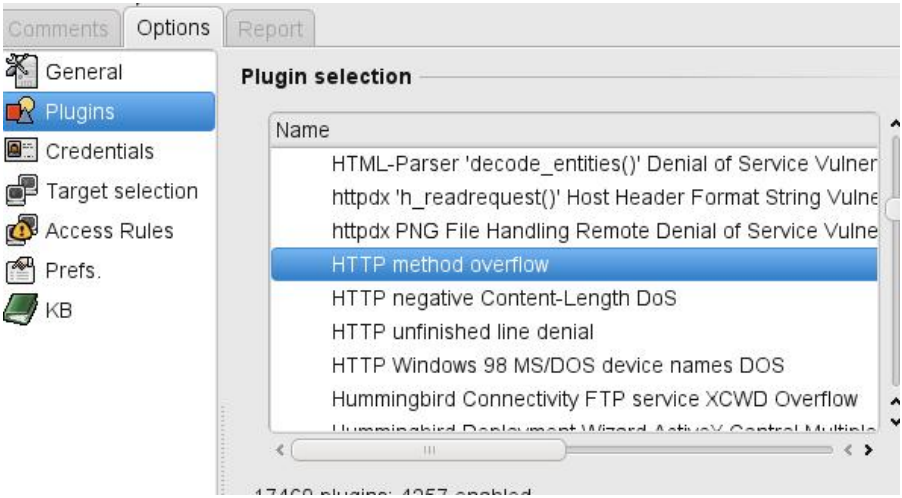
Es folgt der nächste Schritt, bei dem Nessus Informationen von jedem einzelnen Host und Dienst sammelt. Für diese Aufgabe sind die Plug-ins der Kategorie *ACT_GATHER_INFO* zuständig. Diese Plug-ins sind von Natur aus harmlos und führen zu keinerlei Beeinträchtigung auf den Zielen. Ihre wichtigste Aufgabe ist das Abfragen von Diensten und die Remote-Analyse.

Die Dienstabfragen sind so konzipiert, dass sie durch passive Checks keine Gefahr für die Ziele darstellen. Zuvor müssen allerdings die Informationen über die Ports verfügbar sein, weil verschiedene Dienste auch über mehrere Ports verfügbar sein

können. So ist der HTTP-Dienst zwar standardmäßig über den Port 80 verfügbar, kann aber auch über weitere Schnittstellen wie 8080, angeboten werden. Diese Informationen werden, wie bereits erwähnt, von den Informationssammlern benötigt. Die Informationen werden dann in die Knowledge Base geschrieben und von den Verwundbarkeits- und DoS-Tests genutzt.

Im Anschluss daran werden jene Plug-ins ausgeführt, die die Ziele auf ihre tatsächlichen Verwundbarkeiten hin untersuchen. Dabei kommen die Plug-ins der Kategorie *ACT_ATTACK*, *ACT_MIXED_ATTACK* und *ACT_DESTRUCTIVE_ATTACK* zum Einsatz. Sie werden auch in dieser Reihenfolge ausgeführt. Im Unterschied zu den informativen Plug-ins können sie durchaus auch Ziele lahm legen.

Während die ACT_ATTACK-Plug-ins gezielt Schwachstellen ansprechen, kommen die ACT_MIXED_ATTACK-Varianten in Verbindung mit der Option *Safe Check* zum Einsatz und liefern diese zusätzlichen Informationen zur angesprochenen Lücke. Die Plug-ins der Kategorie *ACT_DESTRUCTIVE_ATTACK* scheren sich nicht um diese Dienste und nehmen die Lücken gezielt unter Beschuss, auch wenn der jeweilige Dienst abstürzen sollte.



Verschiedene Skripts, wie das *HTTP method overflow*-Plug-in, können die Zielsysteme auch zum Abstürzen bringen.

7.2 Knowledge Base

Durch die Kategorisierung der Plug-ins wird eine strenge Aufgabentrennung in OpenVAS erreicht. Die Testergebnisse landen allesamt in der Knowledge Base. Die Ergebnisse einzelner Hosts werden in einem speziellen Knowledge Base-Verzeichnis abgelegt. Hat der Benutzer *User* beispielsweise Host *192.168.0.2* gescannt, so findet man die zugehörigen Knowledge Base-Einträge bei einer Standardinstallation unter */var/lib/openvas/users/User/kbs/192.168.0.2*. Es handelt sich um eine textbasierte Sammlung der durchgeführten Tests. Nachfolgend ist ein Ausschnitt aus der Wissensdatenbank abgebildet:

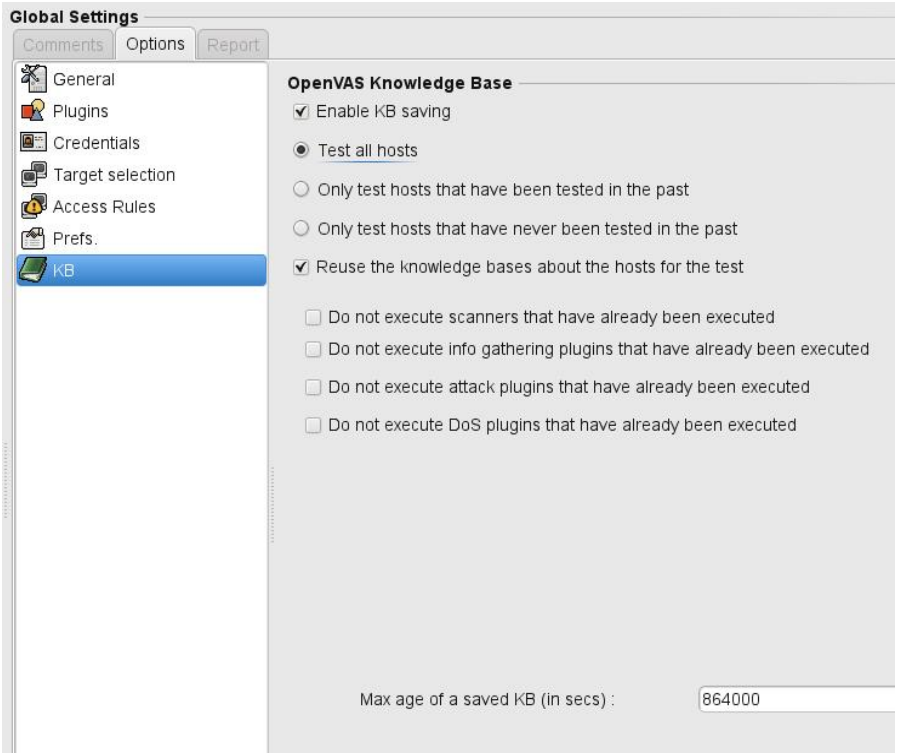
```
1098251953 3 Launched/12288=1
1098251953 1 global_settings/experimental_scripts=no
1098251953 1 global_settings/thorough_tests=no
1098251953 1 global_settings/report_verbosity=Normal
1098251953 1 global_settings/log_verbosity=Normal
1098251953 1 global_settings/debug_level=0
1098251953 3 Launched/10180=1
1098251953 3 Launched/10335=1
1098251953 1 Ports/tcp/23=1
1098251953 1 Ports/tcp/22=1
1098251953 1 Ports/tcp/21=1
1098251953 1 Ports/tcp/25=1
1098252103 1 Ports/tcp/80=1
1098252153 1 Ports/tcp/110=1
1098252203 1 Ports/tcp/139=1
1098252203 1 Ports/tcp/135=1
1098252853 1 Ports/tcp/445=1
1098253053 1 Ports/tcp/554=1
1098254063 1 Host/scanned=1
1098254063 3 Launched/10890=1
1098254063 3 Launched/10870=1
1098254063 3 Launched/10308=1
1098254063 3 Launched/10889=1
```

```
1098254063 3 Launched/11933=1
1098254063 3 Launched/10917=1
1098254063 3 Launched/10274=1
1098254063 3 Launched/11203=1
1098254063 1 SMTP/headers/From=nobody@example.com
1098254063 1 SMTP/headers/To=postmaster@[192.168.0.2]
1098254063 1 Set-tings/third_party_domain=example.com
1098254063 1 ftp/writeable_dir=/incoming
1098254063 1 ftp/login=anonymous
…
```

Die Inhalte sind natürlich von den durchgeführten Tests abhängig. Während manche Zeilen rein informative Daten enthalten, beinhalten andere Scan-Einstellungen oder Details zu Reaktionen auf Scan- und Testvorgänge. Der Knowledge Base kann man exakt entnehmen, welche Daten beispielsweise an einen Dienst gesendet wurden. Auch der Verwundbarkeitstyp lässt sich ihr entnehmen.

Während der Tests wird übrigens eine temporäre Datei erzeugt, in der die aktuellen Ergebnisse landen. Es handelt sich um eine Datei im NBE-Format, die standardmäßig im TMP-Verzeichnis erzeugt wird. Nachdem die Tests abgeschlossen sind, kann man die Daten wie oben beschrieben in eines der vielen Exportformate konvertieren.

Um in den Genuss der Knowledge Base-Daten zu gelangen, müssen Sie die Knowledge Base zunächst aktivieren. Dazu öffnen Sie in den Scan-Einstellungen das Register *KB* und aktivieren die Wissensbasis, indem Sie das Kontrollkästchen *Enable KB saving* anklicken.



Die Konfiguration der Knowledge Base im OpenVAS-Client.

Die Daten, die die Knowledge Base verwaltet, werden aufseiten des OpenVAS-Servers verwaltet. Zuvor wurde bereits darauf hingewiesen, dass für einzelne Hosts eigene Dateien erzeugt werden und diese bei einer Standardinstallation unter */var/lib/openvas/users/<USERNAME>/kbs* abgelegt sind. Mit folgendem Kommando kann man die Inhalte des KB-Verzeichnisses abrufen:

```
ls /var/lib/openvas/users/<USERNAME>/kbs
```

Je nach Umfang der durchgeführten Tests werden die IP-Adressen bzw. die Hostnamen der gescannten Ziele ausgegeben, beispielsweise:

```
Scanner IP : 192.168.1.4
Port scanner(s) : openvas_tcp_scanner
Port range : default
```

```
Thorough tests : no

Experimental tests : no

Paranoia level : 1

Report Verbosity : 1

Safe checks : no

Max hosts : 20

Max checks : 4

Scan Start Date : 2010/5/27 11:13

Scan duration : 193 sec

1274951788 3 Success/1.3.6.1.4.1.25623.1.0.19506=1

1274951794 1 SentData/1.3.6.1.4.1.25623.1.0.810002/LOG=No CPE
identities could be determined.

1274951794 3 Success/1.3.6.1.4.1.25623.1.0.810002=1

1274951794 3 Launched/1.3.6.1.4.1.25623.1.0.100353=1

1274951795 1
SentData/1.3.6.1.4.1.25623.1.0.95000/NOTE=Prüfergebnisse ge-
mäß IT-Grundschutz, 10. Ergänzungslieferung:

IT-Grundschutz 4.1 Passwortschutz für IT-Systeme

Ergebnis: Prüfroutine für diese Maßnahme ist nicht verfügbar.

Details: Prüfroutine für diese Maßnahme ist nicht verfügbar.

IT-Grundschutz 4.2 Bildschirmsperre

Ergebnis: Prüfroutine für diese Maßnahme ist nicht verfügbar.

Details:     Prüfroutine für diese Maßnahme ist nicht verfüg-
bar.

IT-Grundschutz 4.3 Regelmäßiger Einsatz eines Anti-Viren-
Programms

Ergebnis:    Prüfroutine für diese Maßnahme ist nicht verfüg-
bar.

Details:     Prüfroutine für diese Maßnahme ist nicht verfüg-
bar.
```

Den Inhalt können Sie natürlich auch mit einem Dateimanager abrufen. Die Konfiguration der Knowledge Base erfolgt über den OpenVAS-Client. Zunächst sollte sichergestellt sein, dass die Option *Enable KB saving* aktiviert ist. Über die folgenden Schalter kann man exakt steuern, welche Hosts unter die Lupe genommen werden und ob die Inhalte der Knowledge Base wieder verwendet werden sollen oder nicht.

Wenn Sie den OpenVAS-Client benutzen, so sollten Sie die Option *Test all hosts* aktivieren, denn so werden alle Systeme, die man als Ziele unter *Target Selection* angegeben hat, unter die Lupe genommen. Alternativ können Sie bei diesem Client die Option *Only test hosts that have been tested in the past* verwenden, um nur bereits getestete Hosts, die Sie schon einmal gescannt haben, zu testen, oder die Option *Only test hosts that have never been tested in the past*, um bislang noch nicht getestete Systeme zu untersuchen.

Der gezielte Einsatz dieser Schalter ist beispielsweise sinnvoll, wenn Sie nach Änderungen einer Subnetz-Umgebung lediglich neu hinzugekommene Systeme untersuchen wollen. Sinnvoll ist deren Verwendung auch, wenn man bei bereits gescannten Umgebungen Sicherheitslücken durch das Einspielen von Patches etc. geschlossen hat und nun überprüfen will, ob sich deren Einsatz positiv auf die Ergebnisse auswirkt. Auch bei der Verwendung von DHCP im lokalen Netzwerk kann es sinnvoll sein, Tests beispielsweise bei allen Hosts durchzuführen. Aktiviert man die Option *Reuse the knowledge bases about the hosts for the test*, so stehen fünf bislang grau hinterlegte Optionen zur Verfügung.

Die ersten vier Optionen korrespondieren zu folgenden vier Plug-in-Kategorien und deren Aktivierung:

- *ACT_SCANNER*
- *ACT_GATHER_INFO*
- *ACT_MIXED_ATTACK, ACT_DESTRUCTIVE_ATTACK, ACT_ATTACK*
- *ACT_DENIAL, ACT_KILL_HOST, ACT_FLOOD*

Dabei handelt es sich nicht um alternative Schalter, sondern die Kategorien können auch zusammen aktiviert werden. Dem aufmerksamen NASL-Kenner wird womöglich aufgefallen sein, dass zwei Kategorien nicht auftauchen: *ACT_INIT* und *ACT_SETTINGS*. Bei beiden handelt es sich um Kategorien für globale Einstellungen. Da sie nicht mit dem System interagieren, können sie auch nicht für die Verwendung der Knowledge Base deaktiviert werden.

Unter *Max. age of a saved KB (in secs)* legt man fest, wie lange die Informationen, die in einer Knowledge Base gespeichert sind, gültig bleiben. Der Standardwert ist 864.000 Sekunden, also umgerechnet zehn Tage. Nach Ablauf dieser Zeit werden die gespeicherten Daten als obsolet behandelt. Es versteht sich von selbst, dass ein Hochsetzen keine besseren Informationen liefert. Wenn man über Änderungen des Standardwerts nachdenkt, sollte man diesen herabsetzen. Gerade bei Netzwerken, die sich in der Entstehungs- und/oder Umbauphase befinden, kann es sinnvoll sein, diesen beispielsweise auf zwei Tage herunterzusetzen.

Bei der Wiederverwendung von Daten, die bereits in der Knowledge Base gespeichert sind, sollten Sie einige Dinge beachten. Unterbinden Sie das erneute Scannen von Diensten, so werden natürlich auch nach dem letzten Scan freigegebene bzw. aktivierte Dienste beim Scan eines Diensttyps nicht durchgeführt, und daher können auch keine Verwundbarkeiten aufgedeckt werden. Im ungünstigsten Fall entgehen so relevante Informationen. Auch die Lebensdauer von Daten sollten Sie nicht unnötig hoch setzen.

Ein weiteres sicherheitstechnisches Problem: Da in der Knowledge Base in der Regel kritische Informationen enthalten sind, sollten Sie sicherstellen, dass die im KB-Verzeichnis abgelegten Dateien nicht für unberechtigte Benutzer zugänglich sind. OpenVAS beschränkt den Zugriff standardmäßig auf den User Root. Sie tun gut daran, diese Einstellung zu verifizieren.

Die wichtigste Aufgabe der Knowledge Base ist das Verwalten von Informationen, die von den unzähligen Plug-ins genutzt werden können. Eine wichtige Aufgabe ist es, Redundanz zu verhindern. Für ein besseres Verständnis der Funktionsweise von Nessus ist es sinnvoll, dass man versteht, wie die Daten von den Skripts genutzt werden. Insbesondere das Herauslesen von Informationen und das Schreiben von Daten in die Knowledge Base sind interessant.

Führen Sie an einem Host umfangreiche Tests durch, so umfasst die zugehörige Datenbank schnell mehrere Hundert Zeilen. Mithilfe von *grep* ist es beispielsweise recht einfach, die Port-spezifischen Informationen aufzuspüren. Hier ein typisches Beispiel für die Ausgabe der Informationen aus der Knowledge Base mittels *grep*:

```
# grep "Port" 192.168.0.2
1098251953 1 Ports/tcp/23=1
1098251953 1 Ports/tcp/22=1
1098251953 1 Ports/tcp/21=1
1098251953 1 Ports/tcp/25=1
1098252103 1 Ports/tcp/80=1
1098252153 1 Ports/tcp/110=1
```

```
1098252203 1 Ports/tcp/139=1
1098252203 1 Ports/tcp/135=1
1098252853 1 Ports/tcp/445=1
1098253053 1 Ports/tcp/554=1
```

OpenVAS versucht mit einem Skript wie beispielsweise *find_service.nasl* herauszufinden, welche Dienste sich hinter den offenen Ports befinden. Gelingt es dem Skript, diese Informationen auszulesen, so schreibt es zwei Informationen in die Host-Knowlegde Base, wobei *port* den Port und *proto* das Protokoll angibt:

```
Known/tcp/port = proto
Services/proto/
```

Für das Schreiben von Dienstinformationen in die Knowledge Base verwendet das Skript das Kommando *register_service*. Hier ein Beispiel für ein entsprechendes Kommando:

```
# KB wird um neue Info erweitert
register_service(port:xxx,proto:yyy);
```

Liest man mithilfe von *grep* wieder die portspezifischen Informationen aus der Knowledge Base aus, so findet man in der Ausgabe die neuen Infos:

```
1098252293 1 Known/tcp/1234=www
1098252293 1 Service/OpenVAS=1234
1098252283 1 Services/yyy=xxx
1098252303 1 Known/tcp/1234=www
1098252303 1 Service/OpenVAS=1234
```

Hier ein Beispiel für ein einfaches Skript, das das Schreiben in die Knowledge Base demonstriert:

```
# Demo fürs Schreiben in die KB


if(description)
{
```

```
 script_id(123456);
 script_version ("$Revision: 1.1$; ");
 name["english"] = "demo"

 desc["english"] =  "
 Dieses Beispiel-Skript schreibt
 Einfach nur in die KB.
 Risk factor : Info";

 script_description(english:desc["english"]);
 summary["english" = "Schreibt in die KB";
 script_summary(english:summary["english"]);
 script_category(ACT_GATHER_INFO);


 script_copyright(English: "Copyrightinfo");
 family["english"] = "Misc. ";
 script_family(english:family["english"]);
 script_dependencie("find_services.nes");
 exit(0)
}
set_kb_item(name: "Problem/test/schlüssel, va-
lue:string("test"));
```

Wie Sie obigem Beispiel entnehmen können, erfolgt das Setzen von Daten mit dem Kommando *set_kb_item*. Um Informationen auszulesen, verwenden Sie *get_kb_item* bzw. *get_kb_list*. Um Abhängigkeiten von anderen Skripts zu setzen, verwenden Sie beispielsweise das Kommando *script_dependencies*. Folgendes Beispiel setzt vor der eigenen Ausführung die Ausführung des Skripts *vorher.nasl* voraus:

```
script_dependencies(„vorher.nasl“);
```

Wie wir in Kapitel 3.5.2 gesehen haben, sind die Skript-Details und auch die Abhängigkeiten über die Skript-Details mit einem Klick auf *Show dependencies* verfügbar.

8 Tipps und Tricks für den Praxiseinsatz

Meist kommt OpenVAS nicht zum Scannen einzelner Hosts, sondern für die Analyse einer bestehenden Netzwerkarchitektur zum Einsatz. Dass das Scannen komplizierter wird, je größer und heterogener die Umgebung ist, versteht sich von selbst. Der Schlüssel für erfolgreiches Scannen eines Firmennetzes ist eine gute Planung und Vorbereitung der Tests. Nicht minder wichtig ist eine effektive Konfiguration der Scanner.

8.1 Planung

Ein wichtiger Schritt für den erfolgreichen Einsatz von OpenVAS ist eine gute Planung. Dass das blinde Scannen aller Hosts, die sich im lokalen Netzwerk befinden, nicht notwendigerweise eine brauchbare Datenbasis liefert, versteht sich bei der Fülle an Informationen, die Nessus ausgibt, von selbst.

Bei einem „normalen“ Netzwerk kann man davon ausgehen, dass pro Host etwa eine Handvoll ernste und eben so viele mittlere Sicherheitslücken bzw. -risiken bestehen. Hinzu kommen meist rund ein Dutzend weniger kritische Verwundbarkeiten. Pro Host kommt man meist auf rund 20 Schwachstellen, denen man sich widmen sollte. Bei einem Netz mit 10 Rechnern kommt man dann auf gut 200 Lücken, bei 20 Rechnern entsprechend auf 400.

Nun steht man oftmals vor der schwierigen Entscheidung, welchen Hosts man sich zuerst widmen sollte. Prinzipiell sollte man sich die Frage stellen, welches System die wichtigsten Daten speichert und die wichtigsten Dienste bereitstellt. Dem System oder den Systemen, deren Ausfall den größten finanziellen und zeitlichen Ausfall verursacht, sollte man sich zuerst widmen.

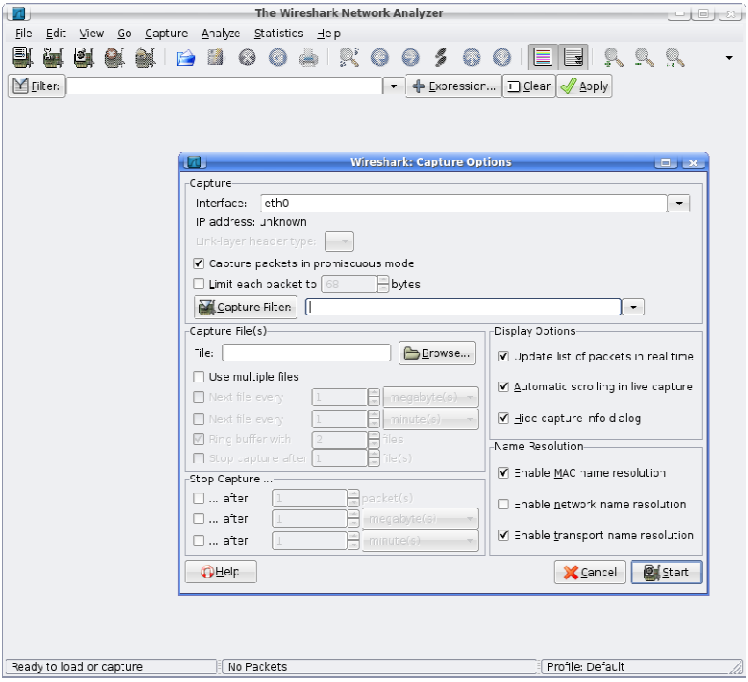
Daher sollten Sie schon vor den ersten Tests eine Prioritätenliste erstellen, die die Bedeutung der Netzwerksysteme widerspiegelt. Diese könnte wie folgt aussehen:

1. Zentrale Server für CMS, Mail, Datenbanken, Außenanbindung etc.
2. Systeme für Kundenverwaltung, Finanzen, etc.
3. Managementsysteme
4. Systeme mit Buchhaltung, Marketing- und Verkaufsdaten
5. Sonstige Clients und Systeme

Wichtig ist natürlich, dass Sie eine eigene Prioritätenliste erstellen. Ergänzend dazu sollten Sie einen Plan der Netzwerkarchitektur erstellen. Dabei sollten Sie für alle Netzwerksysteme Hostname, IP-Adresse, Funktion, Anwender, wichtige Systeminformationen wie Ausstattung und Software-Installationen dokumentieren.

Mit diesen Informationen fällt die weitere Planung leichter. So können Sie beispielsweise entscheiden, welche Systeme wie oft unter die Lupe genommen werden müssen. Bei komplexen Netzen, in denen beispielsweise verschiedene Subnetze von unterschiedlichen Administratoren betreut werden, müssen außerdem Zuständigkeiten geklärt werden.

Wichtig bei Netzen mit Außenanbindung: Die interne Netzwerktopologie sollte von innen, die externe Topologie von außen gescannt werden. Nur so erhalten Sie ein exaktes Bild der Sicherheitsrisiken von innen wie auch von außen. Für ein kleines Netzwerk sind daher zumindest zwei OpenVAS-Server erforderlich, die die Tests durchführen.

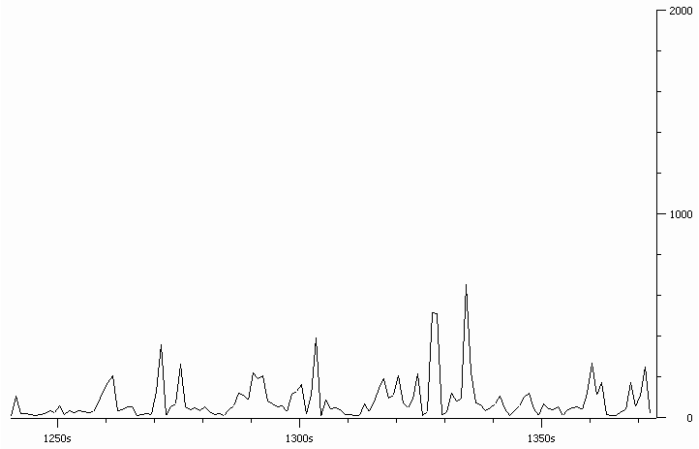


Der Open-Source-Sniffer Wireshark ist für Windows und Linux verfügbar. Er erlaubt das Aufzeichnen und Überwachen des Netzwerk-Traffics.

8.2 Bandbreite

Mit einem weiteren wichtigen Aspekt muss man sich auseinandersetzen: Die erforderliche Bandbreite. Sicherheitschecks, wie sie von Nessus durchgeführt werden, führen zu einer erheblichen Belastung des Netzwerks. Um die tatsächliche Belastung durch den Scanner einschätzen zu können, benötigen Sie fundierte Informationen, die zwei gängige Programme liefern: tcpdump (*http://www.tcpdump.org*) und Wireshark (*http://www.wireshark.org*).

tcpdump ist ein Netzwerk-Sniffer, der als Netzwerk-Monitor und/oder Diagnoseprogramm zum Einsatz kommt. Da die Informationen, die tcpdump produziert, von Natur aus schwer zu lesen sind, greift man zu Wireshark. Wireshark ist ein kostenloser Netzwerkprotokoll-Analysator. Er bietet die Möglichkeit, Daten im laufenden Netzwerkbetrieb zu analysieren und zudem gesammelte Daten aus einem Capture-File zu lesen und auszuwerten. Es können auch Details zu jedem einzelnen TCP/IP-Paket angezeigt werden. Zusätzlich verfügt Wireshark über leistungsfähige Analyse- und Summary-Funktionen sowie eine eigene Sprache zum Filtern von Netzwerkpaketen.



Ein Beispiel für die grafische Aufbereitung des Netzwerk-Traffics im lokalen Netzwerk.

Wireshark kann nicht nur den Netzwerkverkehr überwachen und in Echtzeit darstellen, sondern diesen auch grafisch darstellen. Dabei kann er für jeden einzelnen Scan Grafiken ausgeben, anhand derer man ablesen kann, welcher Traffic durch

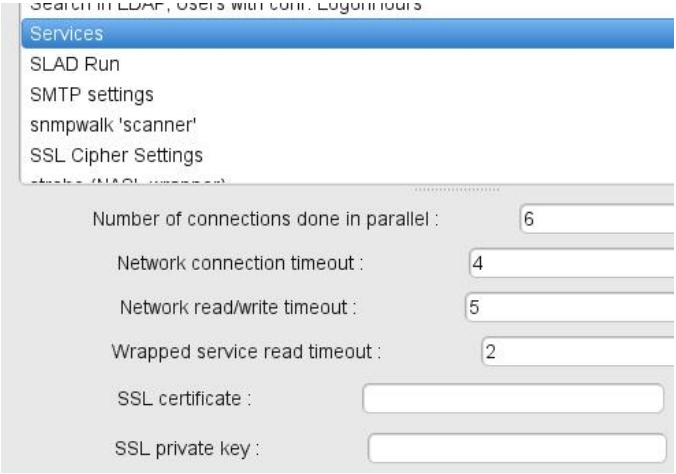
einen bestimmten Scan-Typ verursacht wird. Über *Statistics> Summary* erhält man außerdem eine detaillierte Zusammenfassung.

Tipp

Kommt OpenVAS regelmäßig im lokalen Netzwerk zum Einsatz, so ist es wichtig, dass die Skripts immer auf dem neuesten Stand sind. Hierfür steht Ihnen das oben beschriebene Update-Skript zur Verfügung, das dafür sorgt, dass Ihre Plug-ins stets aktuell sind.

Kann man den durch die OpenVAS-Scans verursachten Traffic abschätzen, so gilt es zu entscheiden, wann OpenVAS seine Arbeit aufnehmen soll, um die Arbeit der Netzwerkumgebung nicht zu beeinträchtigen.

Um den OpenVAS-Traffic zu minimieren, beschränken Sie einfach die Anzahl der Scans, die gleichzeitig durchgeführt werden. Dazu passen Sie die Anzahl der gleichzeitigen Scans auf dem Register *Prefs* des OpenVAS-Clients an. Durch ein Heruntersetzen vom Standardwert auf 4 verringert sich der Traffic entsprechend. Allerdings dauert der gesamte Scan-Vorgang auch entsprechend länger. Außerdem kann man Scans auf mehrere Server verteilen und auch zeitlich steuern. Bei sehr umfangreichen Tests ist es außerdem sinnvoll, diese per Cronjob außerhalb der Arbeitszeit automatisiert durchzuführen.



Durch den gezielten Einsatz der Scan-Optionen kann der Netzwerk-Traffic spürbar verringert werden.

Ist geklärt, welche Systeme mit welcher Priorität und mit welchen Testskripts untersucht werden sollen, sollten Sie festlegen, ob ein oder auch mehrere OpenVAS-Installationen das Netz unter die Lupe nehmen. Besteht eine Außenanbindung, so sollte das Netz, wie bereits erwähnt, von innen wie auch von außen analysiert werden. Bei komplexen Netzwerkstrukturen ist es außerdem sinnvoll, die Tests auf Arbeitsgruppen und/oder Abteilungen zu beschränken.

Achtung: Hoher Speicherbedarf!

Der OpenVAS-Daemon ist von Haus aus ein recht genügsames Werkzeug. Er benötigt gerade mal 1 MB Arbeitsspeicher. Dieser Bedarf ist allerdings von der Anzahl der Plug-ins abhängig, die gleichzeitig ausgeführt werden. Beim Scannen eines Hosts werden bereits bei einem Plug-in 10 MB Arbeitsspeicher benötigt, bei 4 Plug-ins sind es 16 MB. Setzt man den maximalen Wert für das gleichzeitige Ausführen von Plug-ins auf 64, so werden durchschnittlich 60 MB benötigt, die Spitzenbeanspruchung kann allerdings auch nahe 100 MB liegen.

8.3 Hurra: keine Detached Scans mehr

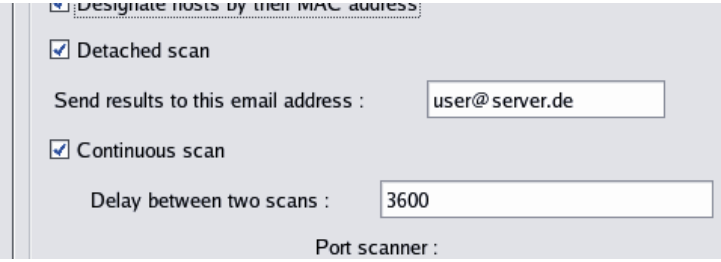
Wenn Sie den OpenVAS-Vorläufer Nessus 2.x kennen, wundern Sie sich womöglich, dass das Stichwort Detached Scans noch nicht gefallen ist. Der Grund hierfür ist einfach: Die Funktionalität, den Server Scans ausführen zu lassen, ohne dass eine Verbindung zu einem Client besteht, wurde aufgrund von Designentscheidungen aufgegeben.

Das hatte zur Folge, dass die Kommandos *DETACHED_SESSIONS_LIST* und *DETACHED_STOP* vom Protokoll entfernt wurden. Das wiederum hatte zur Folge, dass damit zusammenhängende Voreinstellungen entfernt wurden, und zwar folgende:

- detached_scan
- continuous_scan
- delay_between_scan_loops
- detached_scan_email_address

Bei Nessus 2.x konnte der Nessus-Server nicht nur in Verbindung mit dem Nessus-Client, sondern auch alleinstehend ausgeführt werden. Man spricht in diesem Zusammenhang auch von sogenannten Detached Scans. Beim dem OpenVAS-Vorläufer lies sich diese Funktion nutzen.

Dazu musste beim Nessus-Client 2.x die zugehörige Funktion über das Register *Scan options* aktiviert sein. War sie aktiv, konnte der Server unabhängig vom Client seine Tests durchführen.



Die Aktivierung des Standalone-Scanners bei Nessus 2.x.

8.4 „Lokale“ Tests

Mit OpenVAS sind auch „lokale“ Sicherheits-Checks möglich. Dabei kann sich der Security-Scanner per SSH Zugang zum System verschaffen und feststellen, welche Patches fehlen. Verfügt der Client über einen gültigen SSH-Schlüssel, so loggt er sich in dem entfernten Host ein, nimmt die Liste der installierten Software unter die Lupe und gibt dann einen Bericht aus, welche Systemkomponenten erneuert werden sollten.

Das Ziel dieser Funktion: Alle fehlenden Patches sollen gefunden werden, also nicht nur die für die externen Dienste, Ports etc., sondern auch die, die für die sichere Ausführung der Umgebung erforderlich sind. Wichtig für das Verständnis: Der lokale Check kann nicht fehlerhafte Benutzerberechtigungen oder Ähnliches aufdecken.

Das Interessante an der Patch-Ermittlung: Es wird schon jetzt eine Vielzahl an Betriebssystemen unterstützt, insbesondere:

- AIX
- Debian
- Fedora Core
- FreeBSD
- Gentoo Linux

- Mac OS X und Mac OS X Server
- Mandrake Linux
- RedHat Enterprise Linux
- Solaris
- SuSE Linux
- Microsoft Windows NT, 2000, XP, 2003

Um den internen Check nutzen zu können, muss zunächst ein SSH-Schlüsselpaar erzeugt werden. Dann erzeugen Sie auf jedem System, dessen Innenleben unter die Lupe genommen werden soll, einen lokalen Account, der für die Tests verwendet wird. Dann kopiert man den öffentlichen SSH-Schlüssel, den OpenVAS verwendet, in das Verzeichnis des neuen Benutzers. Abschließend teilt man OpenVAS mit, welche Schlüssel verwendet werden sollen.

Um einen öffentlichen SSH-Schlüssel zu erzeugen, verwenden Sie den Befehl *ssh-keygen*:

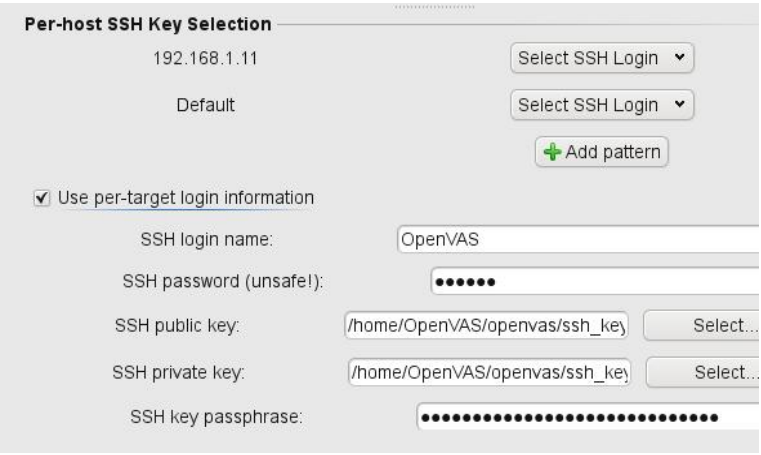
```
$ ssh-keygen -t dsa

Generating public/private dsa key pair.

Enter file in which to save the key
(/users/user/.ssh/id_dsa): /home/user/openvas/ssh_key

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in
/home/user/openvas/ssh_key.

Your public key has been saved in
/home/user/openvas/ssh_key.pub.

The key fingerprint is:

4d6:5a:sd:88:gg:1g:d4:e0:7b:89:98:9i:88:r6:12:ea
user@server.local
```

Wie Sie obiger Ausgabe entnehmen können, werden zwei Dateien erzeugt:

- **/home/user/openvas/ssh_key**: Das ist der private SSH-Schlüssel.
- **/home/user/openvas/ssh_key.pub**: Das ist der öffentliche SSH-Schlüssel.

Nun erzeugen Sie einen SSH-Benutzeraccount mithilfe von *adduser*. Als Nächstes kopieren Sie den Schlüssel in dessen Home-Verzeichnis und setzen die korrekten Berechtigungen für das SSH-Verzeichnis:

```
# mkdir ~openvas/.ssh
# mv ssh_key.pub ~openvas/.ssh/authorized_keys
# chown -R openvas:openvas ~openvas/.ssh/
# chmod 0600 ~ openvas/.ssh/authorized_keys
# chmod 0700 ~ openvas/.ssh/
```



Das lokale Testen setzt die korrekte Konfiguration von SSH voraus.

Abschließend müssen Sie nur noch Nessus entsprechend konfigurieren. Dazu öffnen Sie die Scan-Einstellungen, wechseln zum Register *Credentials* und tragen im Menü *SSH Authorization* folgende Informationen ein:

- SSH-Benutzername
- SSH-Passwort
- Verzeichnis des öffentlichen SSH-Schlüssels
- Verzeichnis des privaten SSH-Schlüssels
- Passwortphrase für den SSH-Schlüssel

Über die Plug-in-Auswahl können Sie natürlich einsehen, welche lokalen Scans verfügbar sind. Die Kategorien tragen beispielsweise die Bezeichnung *Fedora Local Security Checks* oder *SuSE Local Security Checks*. Wie gewohnt startet man die Tests über die Schaltfläche *Scan*. Der Berichtausgabe können Sie dann entnehmen, dass sich Nessus per SSH eingeloggt hat. Obige Schritte muss man für jedes System wiederholen, das Sie einem internen Check unterziehen wollen.

8.5 Scannen von Windows-Systemen

OpenVAS wird häufig auch in heterogenen Umgebungen zum Scannen von Windows-Hosts verwendet. Auch bei Windows-Systemen ist ein Scannen des Innenlebens, genauer der Windows-Registrierungsdatenbank möglich. Diesen Zugriff haben in der Regel nur Windows-Systemadministratoren. Aber mit einigen Eingriffen in ein Windows-System lässt sich auch dies mittels Nessus analysieren.

Dazu sind allerdings auch Eingriffe in die Windows-Registrierungsdatenbank erforderlich. Werden diese nicht sachgemäß durchgeführt, kann man sich eine komplette Windows-Installation zerschießen. Daher sollte man zumindest die Registrierungsdatenbank vor den Eingriffen sichern.

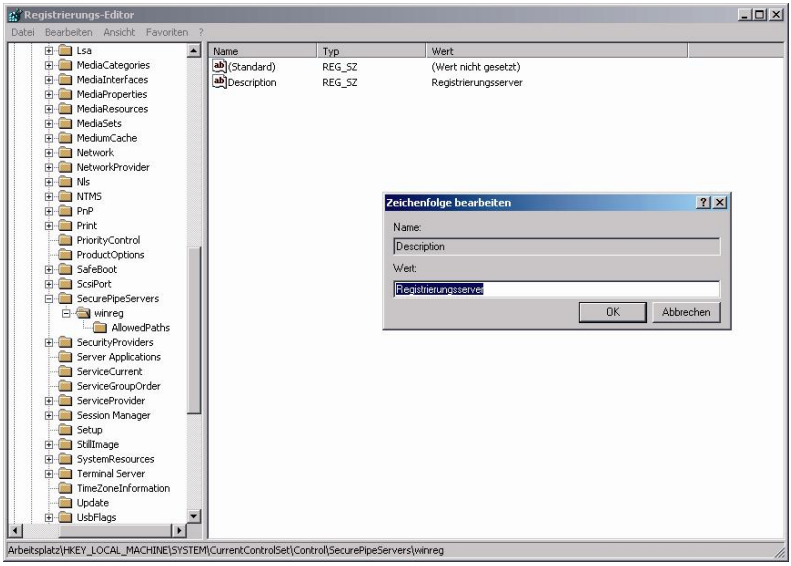
Zunächst sind eine ganze Reihe von Anpassungen aufseiten des Windows-Systems erforderlich. Als Erstes erzeugen Sie eine neue Benutzergruppe, die man beispielsweise als *OpenVAS-Test* bezeichnet. Als Nächstes erzeugen Sie in dieser Benutzergruppe einen OpenVAS-User, den Sie beispielsweise einfach als *OpenVAS* bezeichnen.

Damit OpenVAS Remote-Zugriff auf das Windows-System erlangen kann, müssen Sie einen Registrierungsschlüssel anpassen. Der Remote-Zugriff wird über folgenden Schlüssel aktiviert:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurePip
eServers\winreg
```

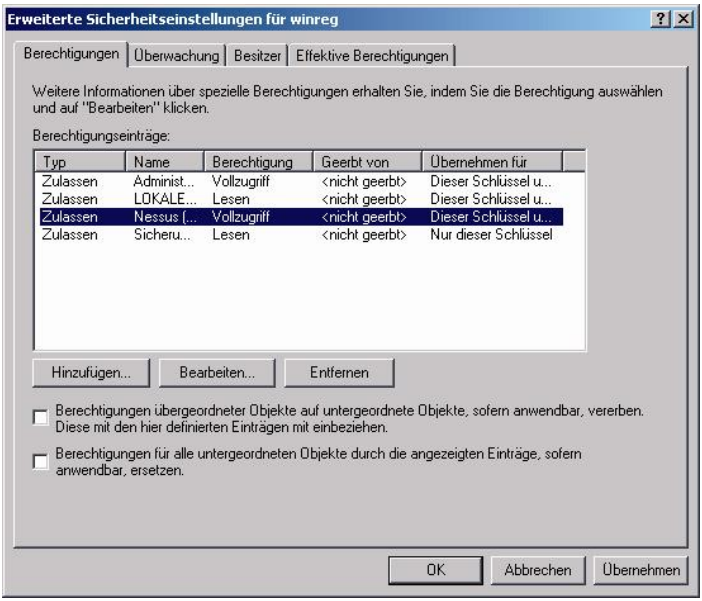
Dieser Schlüssel muss nun angepasst werden, damit der zuvor erzeugte OpenVAS-Benutzer auf das System zugreifen kann. Dazu führen Sie folgende Schritte aus:

1. Zunächst öffnen Sie mit dem Registrierungseditor den Schlüssel *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control.*
2. Diesen markieren Sie, führen dann den Befehl *Neu> Schlüssel* aus und bezeichnen diesen als *SecurePipeServers*. Als Typ verwenden Sie *REG_SZ*.



Durch Eingriffe in die Windows-Registrierungsdatenbank verschaffen Sie sich Zugang zu Windows-Systemen.

3. Dann öffnen Sie den neu erzeugten Schlüssel *SecurePipeServers* und erzeugen einen neuen Unterschlüssel, den Sie mit *winreg* bezeichnen.
4. Außerdem erzeugen Sie eine neue Zeichenfolge, bezeichnen diese mit *Description* und geben als Wert *REG_SZ* an.
5. Abschließend modifizieren Sie die Zeichenfolge *Description* und weisen ihr den Wert *Registrierungsserver* zu.



Die Berechtigungen für den OpenVAS-Benutzer.

Sollte der Schlüssel *SecurePipeServers* bereits bestehen, so sparen Sie sich die zuvor beschriebenen Schritte. Als Nächstes gilt es, die Berechtigungen für die Zusammenarbeit mit OpenVAS zu setzen. Auch hierfür nutzen Sie den Registrierungseditor. Mit diesem führen Sie folgende Schritte aus.

1. Zunächst öffnen Sie den oben erzeugten Schlüssel und führen dann den Befehl *Bearbeiten> Berechtigungen* aus.
2. Im Dialog *Berechtigungen* für *winreg* klicken Sie auf *Erweitert*, um die erweiterten Berechtigungen zu öffnen.
3. Dann klicken Sie auf *Hinzufügen*, um den OpenVAS-Benutzer hinzuzufügen und bestätigen mit *OK*.
4. Auf dem Register *Berechtigungen* bearbeitet man die Einstellungen des OpenVAS-Benutzers *OpenVAS*, der zumindest Leserechte benötigt. Abschließend speichern Sie die Einstellungen.

Als Nächstes müssen nur noch die Einstellungen des OpenVAS-Clients angepasst werden. Dort öffnet man das Register *Credentials* und öffnet die Windows-Credentials-Einstellungen. Unter *SMB account* und *SMB password* geben Sie die oben verwendete Benutzerkennung und das Passwort des erzeugten OpenVAS-

Benutzers ein. Nicht zwingend ist die Angabe der SMB-Domain. Bei der nächsten Ausführung von Test-Skripts, die Zugriff auf die Windows-Registry benötigen, kann OpenVAS auch auf diese zugreifen und die relevanten Informationen auslesen.

8.6 Verteiltes Scannen

OpenVAS eignet sich hervorragend für das Scannen von kleinen bis mittleren Netzwerken. Aber schon beim Scannen von mehreren Subnetzen, etwa von unterschiedlichen Abteilungen, stellt sich das Problem, wie man dies in die Praxis umsetzen soll. Ein Hauptproblem von Security-Scannern ist, dass sie recht langsam sind. Eine nennenswerte Beschleunigung ist nur durch das Herausnehmen von Tests möglich. Doch dabei läuft man immer Gefahr, die wirklich relevanten Sicherheitslücken zu übersehen.

Das Scannen eines Class B-Netzes mit mehr als 60.000 Hosts dauert gut und gerne eine ganze Woche. Ein zu langer Zeitraum, um sicherstellen zu können, dass das Netzwerk nicht nennenswerte sicherheitskritische Änderungen vollzogen hat.

Auch die Netzwerkbelastung derart umfangreicher Tests ist kritisch und kann erhebliche negative Auswirkungen auf die Verfügbarkeit und Performance von wichtigen Netzwerk- und Anwendungsdiensten haben. Ein weiteres Problem: Die Kommunikation zwischen den verschiedenen Netzen kann durch Sicherheitslücken, einen Systemcrash oder vergleichbare Probleme eingeschränkt oder gar unterbrochen sein. Auch Scans können natürlich für den Crash eines Routers oder Proxy-Servers verantwortlich sein.

Die Lösung ist scheinbar einfach: Man verteilt das Scannen auf mehrere Scanner-Instanzen, die im Netzwerk verteilt agieren. Mit diesem Ansatz schlägt man gleich mehrere Fliegen mit einer Klappe. Zum einen werden die Tests, die beispielsweise von einzelnen OpenVAS-Installationen pro Abteilung durchgeführt werden, deutlich schneller durchgeführt. Außerdem ist die Netzwerkbelastung erkennbar geringer und man erhält ein wesentlich exakteres Bild über das gesamte Netzwerk.

Einziges Problem: Mit OpenVAS alleine lässt sich verteiltes Security-Scannen bislang nicht realisieren. Dazu bedarf es Spezialfunktionen, die sich um die Steuerung der verschiedenen OpenVAS-Installationen kümmern. Die gute Nachricht: Das OpenVAS-Team hat unter dem Arbeitstitel Master-Slave-Feature eine entsprechende Funktion angekündigt, die noch 2010 kommen soll.

9 Eigene Tests schreiben

Der Security-Scanner OpenVAS verfügt mit NASL über eine eigene Skriptsprache, auf der die Tests basieren. Der interessierte Anwender und Administrator kann sie nutzen, um mit wenig Aufwand eigene Testskripts zu schreiben. Der Vorteil: Man ist nicht an die vordefinierten Test-Skripts gebunden. Dabei bietet NASL handliche Funktionen, um auf einfache Weise Sicherheitstests für unterschiedliche Server-Typen, insbesondere für Webserver und FTP-Server zu erstellen. NASL garantiert, dass ein NASL-Skript lediglich Pakete zum Zielrechner und zu keinem anderen Rechner sendet und dass keine Befehle auf dem lokalen System ausgeführt werden.

Zu viel sollte man allerdings nicht von NASL erwarten, denn sie dient lediglich dem Erstellen von Skripts für Sicherheitstests. NASL ist im direkten Vergleich zu Perl & Co. zwar deutlich weniger leistungsfähig und außerdem um ein Vielfaches langsamer, dennoch ist NASL sicherer, da sie lediglich auf einen Aufgabenbereich beschränkt ist. NASL beansprucht keine großen Speichermengen. Deshalb ist es möglich, zwanzig OpenVAS-Threads gleichzeitig durchführen zu lassen, ohne ein System gleich in die Knie zu zwingen. Da NASL ein eigenständiges System ist, muss man nicht weitere Pakete für jeden neuen Sicherheitstest installieren.

Der Einsatz von NASL ist aus mehreren Gründen sinnvoll:

- NASL ist optimal auf OpenVAS abgestimmt. Einen Test in dieser Sprache zu schreiben, geht flott.
- NASL hat viele Gemeinsamkeiten mit C.
- NASL erzeugt sichere und einfache Sicherheitstests, die man problemlos an andere Anwender weitergeben kann.
- NASL erzeugt portable und einfach modifizierbare Sicherheitstests.

Man unterscheidet übrigens zwischen NASL1, NASL2 und NASL3. Die zweite Variante bietet neben zusätzlichen Befehlen mehr Performance. Die NASL-Skripts speichert man mit der Dateierweiterung NASL und legt sie im Plug-in-Verzeichnis von Nessus an. Bei einer Red-Hat-Installation findet man die Skripts beispielsweise unter */usr/local/lib/openvas/plugins/*. NASL3 ist nur in Nessus 3.x und seinen Nachfolgern verfügbar.

Bevor wir uns NASL im Detail anschauen, schauen wir uns ein Beispiel an, das die Version eines MySQL-Servers abfragt:

```
##################################################################
# OpenVAS Vulnerability Test
# $Id: mysql_version.nasl 3171 2009-04-23 19:21:19Z mime $
#
# Detection of MySQL
#
# Authors:
# Michael Meyer
#
# Copyright:
# Copyright (c) 2009 Michael Meyer
#
# This program is free software; you can redistribute it
# and/or modify it under the terms of the GNU General Public
# License version 2 (or any later version), as published by
# the Free Software Foundation.
#
# This program is distributed in the hope that it will be
# useful, but WITHOUT ANY WARRANTY; without even the implied
# warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
# PURPOSE.
# See the GNU General Public License for more details.
# You should have received a copy of the GNU General Public
# License along with this program; if not, write to the Free
# Software Foundation, Inc., 51 Franklin St, Fifth Floor,
# Boston, MA 02110-1301 USA.
##################################################################

desc["english"] = "
```

```
Overview:

MySQL, an open source database system is running at this
host.


See also:

http://www.mysql.com


Risk factor : None";


if (description)

{

 script_id(100152);

 script_version ("1.0");


 script_name(english:"MySQL Detection");


 script_description(english:desc["english"]);

 script_summary(english:"Check for MySQL");

 script_category(ACT_GATHER_INFO);

 script_family(english:"Service detection");

 script_copyright(english:"This script is Copyright (C) 2009
Michael Meyer");

 script_dependencie("find_service.nes");

 script_require_ports("Services/mysql", 3306);

 exit(0);

}


include("global_settings.inc");

include("misc_func.inc");


port = get_kb_item("Services/mysql");
```

```
if(!port)port = 3306;


if(!mySQL_version=get_mysql_version(port)) { # I found no
Plugin that ever set mysql_version ("mysql/version/"). But
perhaps i missed somthing, so i check first if version is
set.


 soc = open_sock_tcp (port);

 if (!soc)exit (0);


 buf = recv(socket:soc, length:4);

 if(!buf)exit(0);


 #
http://forge.mysql.com/wiki/MySQL_Internals_ClientServer_Prot
ocol


 plen = ord(buf[0]) + (ord(buf[1])/8) + (ord(buf[2])/16); #
Packet Length

 if(ord(buf[3]) != 0)exit(0); # The first packet of a client
query will have Packet Number = 0


 buf = recv (socket:soc, length:plen);

 if(strlen(buf) != plen)exit(0);


 if(ord(buf[0]) == 255 && "not allowed to connect to this
MySQL" >< buf) { # connect not allowed

  MySQL_FOUND = TRUE;

 }

 else if(ord(buf[0]) == 10) { #  connect allowed

  MySQL_FOUND = TRUE;

  for (i=1; i<strlen(buf); i++) {

   if (ord(buf[i]) != 0) { # server_version is a Null-
Terminated String
```

```
    mySQL_version += buf[i];
   } else {
    break;
   }
  }
 }
} else {
   MySQL_FOUND = TRUE;
   getVERSION = TRUE;
}

if(MySQL_FOUND) {

   txt = desc["english"];

   if(mySQL_version) {
    if(!getVERSION) {
     set_mysql_version(port:port, version:mySQL_version);
    }

    info = string("None\n\nMySQL Version '");
    info += mySQL_version;
    info += string("' was detected on the remote host.\n\n");

    txt = ereg_replace(
       string:desc["english"],
       pattern:"None$",
       replace:info
    );
```

```
  }
  register_service(port:port, proto:"mysql");
  if(report_verbosity > 0) {
    security_note(port:port, data: txt);
  }
  exit(0);
}

exit(0);
```

9.1 NASL-Grundlagen

Die NASL-Syntax ist jener von C sehr ähnlich, mit der Ausnahme, dass eine Vielzahl nicht benötigter Dinge entfernt worden ist. Es muss weder auf Objekttypen geachtet noch Speicher reserviert und freigegeben werden. Es ist auch nicht notwendig, Variablen vor ihrem Gebrauch zu deklarieren. Es genügt, sich auf den zu erstellenden Sicherheitstest zu konzentrieren.

Die NASL-Syntax ist C sehr ähnlich und sollte für alle, die einmal ein wenig programmiert haben, leicht verständlich und ebenso einfach nachzuvollziehen sein. Kommentare werden wie gewohnt mit dem Kommentarzeichen # eingeleitet. Gültige Kommentare sind beispielsweise:

```
a = 1;  # Belege a mit Wert 1


# Belege Feld b mit dem Inhalt 2:
b = 2;
```

Ungültige Kommentare sind beispielsweise die folgenden:

```
#
Set a to 1 :
                #
a = 1;
a = # set a to 1 # 1;
```

Variablen muss man vor ihrem Einsatz nicht deklarieren. Auch Variablentypen muss man keine Beachtung schenken. Der NASL-Interpreter meldet sich, wenn versucht wird, etwas Falsches zu tun, beispielsweise das Hinzufügen einer Zahl an ein IP-Paket. Auch auf Memory Allocation und Includes muss man nicht achten. Es gibt keine Includes. Memory Allocation wird, wenn entsprechender Speicher benötigt wird, automatisch durchgeführt.

OpenVAS unterstützt drei Zahlensysteme: dezimal, hexadezimal oder binär. Folgende Syntax ist korrekt:

```
a = 1204;

b = 0x0A;

c = 0b001010110110;

d = 123 + 0xFF;
```

Zeichenketten müssen quotiert werden. Zu beachten ist außerdem, dass im Gegensatz zu C Buchstaben nicht interpoliert werden, sofern nicht explizit die Anweisung dafür über die string()-Funktion erfolgt:

```
a = "Hallo\nIch bin Herbert";          # a wird auf Wert
"Hallo\nIch bin Herbert" gesetzt


b = string("Hallo\nIch bin Herbert");   # b wird auf Wert
"Hallo\nIch bin Herbert" gesetzt


c = string(a);                         # c wird auf Wert von
a gesetzt
```

NASL unterstützt anonyme und nicht-anonyme Funktionen. Schauen wir uns zunächst die nicht-anonymen Funktionen an. NASL unterscheidet sich von C in der Behandlung von Funktionsargumenten. Bei NASL muss man sich nicht an eine korrekte Reihenfolge der Argumente halten. Man spricht von einer nicht-anonymen Funktion, wenn die Reihenfolge der Argumente einer Funktion wichtig ist und die verschiedenen Funktionsargumente unterschiedlichen Typs sind. Wird ein Element vergessen, so erfolgt während der Laufzeit eine entsprechende Hinweisaufforderung bzw. Fehlermeldung. Dazu ein Beispiel. Die Funktion *forge_ip_packet()* besitzt viele Elemente. Die beiden folgenden Aufrufe sind gültig und rufen exakt dasselbe auf:

```
forge_ip_packet(ip_hl : 5, ip_v : 4,
```

```
            ip_p : IPPROTO_TCP);

forge_ip_packet(ip_p : IPPROTO_TCP,
            ip_v : 4, ip_hl : 5);
```

Dabei wird der User während der Laufzeit auf die fehlenden Argumente hingewiesen. Bei einem Sicherheitstest kann der User nicht direkt mit dem Skript interagieren. Vielmehr ist Interaktion bei der Fehlersuche gefragt. Anonyme Funktionen sind Funktionen, die nur ein Argument oder Argumente desselben Typs verwenden. Beispiele:

```
send_packet(my_packet);
send_packet(packet1, packet2, packet3);
```

Diese Funktionen können Optionen beinhalten. Beispielsweise wartet *send_packet()* auf eine Antwort. Wenn keine Notwendigkeit besteht, die Rechner-Antwort einzulesen, dann kann *pcap* deaktiviert und dadurch der Sicherheitstest beschleunigt werden:

```
send_packet(packet, use_pcap:FALSE);
```

Die Konditionen *For* und *While* funktionieren wie in C. Ein Beispiel für die Verwendung von For:

```
for(instruction_start;condition;end_loop_instruction)
{
 #
 # Hier einige Anweisungen
 #
}
```

oder

```
for(instruction_start;condition;end_loop_instruction)Funktion
();
```

Ein Beispiel für *While*:

```
while(condition)
{
 #
 # Hier einige Anweisungen
 #
}
```

oder

```
while(condition)Funktion();
```

Und noch ein Beispiel:

```
       # Zähle von 1 bis 10
       for(i=1;i<=10;i=i+1)display("i : ", i, "\n");
```

NASL unterstützt auch benutzerdefinierte Funktionen. Eine solche Funktion wird wie folgt definiert:

```
Funktion meine_funktion(argument1, argument2, ....)
```

Benutzerdefinierte Funktionen müssen nicht-anonyme Argumente verwenden. Rekursion wird abgehandelt. Ein Beispiel:

```
function fact(n)
{
  if((n == 0)||(n == 1))
    return(n);
  else
    return(n*fact(n:n-1));
}
display("5! ist ", fact(n:5), "\n");
```

Wichtig ist außerdem, dass benutzerdefinierte Funktionen keine weiteren benutzerdefinierten Funktionen beinhalten dürfen. Bei der Entwicklung von NASL-Skripts sollte man außerdem darauf achten, dass, wenn eine Funktion einen Wert zurückgeben soll – und das ist ja oftmals die Aufgabe einer Funktion – die Funktion *return()* angewendet werden muss. Da *return()* eine Funktion ist, muss der Rückgabewert in Klammern gesetzt werden. Ein Beispiel für eine fehlerhafte Verwendung:

```
function func()
{
   return 1; # Hier fehlen die Klammern!
}
```

NASL unterstützt auch die Standard-Operatoren +, -, *, / und %, die man von C kennt. Die Priorität der Operatoren ist bislang noch nicht definiert. Auch die Binär-Operatoren | und & sind implementiert.

NASL stellt zwei weitere Operatoren zur Verfügung, die C nicht kennt:

- **x-Operator**: Dieser Operator wiederholt dieselbe Funktion n-mal. Ein Beispiel:
  ```
  send_packet(udp) x 10;
  ```
  Bei diesem Beispiel wird das gleiche UDP-Paket zehnmal gesendet.

- ><-**Operator**: Hierbei handelt es sich um einen booleschen Operator. Er liefert den Wert *true*, wenn eine Zeichenkette A eine Zeichenkette B enthält. Ein Beispiel:
  ```
  a = "Nessus";
  b = "Ich verwende Nessus";

  if(a >< b){
         # Wird so lange durchgeführt, bis
         # a in b vorkommt
         display(a, " kommt in ", b, " vor\n");
         }
  ```

9.2 NASL-Netzwerkfunktionen

NASL erlaubt immer nur das Öffnen eines Sockets zu einem vom Anwender angegebenen Ziel. Der Aufbau zu anderen Hosts ist mit NASL nicht möglich. Für das Öffnen eines Sockets verwendet man die Funktionen *open_sock_tcp()* und *open_sock_udp()*. Sie öffnen, wie schnell zu erkennen ist, einen TCP- bzw. einen UDP-Socket. Beide Funktionen verwenden anonyme Argumente. Dabei ist immer nur ein Verbindungsaufbau zu einem Port möglich. Schauen wir uns ein Beispiel an:

```
# Öffnet einen Socket am TCP-Port 80:
soc1 = open_sock_tcp(80);
# Öffnet einen Socket am UDP-Port 123:
soc2 = open_sock_udp(123);
```

Die Funktion *open_sock* gibt den Wert 0 zurück, wenn keine Verbindung zum entfernten Rechner aufgebaut werden kann. Die Funktion *open_sock_udp()* kann in der Regel nicht fehlschlagen, da sich meist nicht feststellen lässt, ob der entfernte UDP-Port offen ist oder nicht. Die Funktion *open_sock_tcp()* hingegen liefert den Rückgabewert 0, wenn der entfernte Port geschlossen ist. Ein einfacher TCP-Port-Scan kann wie folgt aussehen:

```
start = prompt("Erster Port zum Scannen? ");
end   = prompt("Letzter Port zum Scannen? ");

for(i=start;i<end;i=i+1)
{
 soc = open_sock_tcp(i);
 if(soc) {
  display("Port ", i, " ist offen\n");
  close(soc);
 }
}
```

Zum Schließen des Sockets verwendet man die Funktion *close()*. Bevor das Socket geschlossen wird, wird intern die Funktion *shutdown()* durchgeführt. Um Sockets zu schreiben und einzulesen, können folgende Funktionen verwendet werden:

- **recv(socket:<socketname>, length:<length> [,timeout : <timeout>)**: Liest <*length*> Bytes vom Socket <*socketname*>. Diese Funktion kann sowohl für TCP als auch für UDP angewandt werden. Die timeout-Option wird in Sekunden angegeben.
- **recv_line(socket:<socketname>, length:<length> [, timeout: <timeout>])**: Diese Funktion entspricht der Funktion *recv()*, das Lesen wird abgebrochen, sobald die erste *n* (New Line) Zeichenfolge erkannt wird.
- **send(socket:<socket>, data:<data> [, length:<length>])**: Sendet die Daten <*data*> an der Socket <socket>. Das optionale Argument *length* teilt der Funktion mit, lediglich <*length*> Bytes an das Socket zu senden. Ist das Argument nicht gesetzt, werden so lange Daten gesendet, bis ein Null-Charakter erkannt wird.

Jene Funktionen, die Socket-Daten einlesen, haben einen internen Timeout-Wert von fünf Sekunden. Wird der Timeout erreicht, wird FALSE zurückgegeben. Ein typisches Beispiel:

```
# Dieses Beispiel zeigt den FTP Banner eines
# entfernten Rechners:
soc = open_sock_tcp(21);
if(soc)
{
 data = recv_line(socket:soc, length:1024);
 if(data)
 {
  display("Der entfernte FTP Banner ist: \n", data, "\n");
 }
 else
 {
  display("Der entfernte FTP Server scheint tcp-wrapped zu
sein\n");
 }
 close(soc);
}
```

NASL verfügt für die Analyse von FTP- und WWW-Diensten über verschiedene High-Level-Funktionen.

- **ftp_log_in(socket:<soc>, user:<login>, pass:<pass>)**: Diese Funktion versucht, sich am FTP-Server über das zuvor aufgebaute Socket <*soc*> anzumelden. Rückgabewert dieser Funktion ist TRUE, wenn der Anmeldeversuch des Users <*log-in*> mit dem Passwort <*pass*> erfolgreich war, FALSE im Fehlerfall.
- **ftp_get_pasv_port(socket:<soc>)**: Führt einen PASV-Befehl am FTP-Server aus und liefert den Port, um eine Verbindung aufbauen zu können. Das ermöglicht NASL-Skripts das Einlesen von Daten per FTP. Diese Funktion liefert den Wert FALSE, wenn ein Fehler aufgetreten ist.
- **is_cgi_installed(<name>)**: Liefert den Wert TRUE, sofern das CGI mit dem Namen <*name*> am entfernten Webserver installiert ist. Diese Funktion verwendet einen GET-Request. Beginnt <*name*> nicht mit einem Schrägstrich, wird *cgi-bin* als Vorspann eingesetzt. Diese Funktion kann auch angewendet werden, um die Existenz einer Datei zu überprüfen.

Schauen wir uns zwei Beispiele für den Einsatz dieser Funktionen an:

```
#
# WWW
#
 if(is_cgi_installed("/robots.txt")){
       display("Die Datei /robots.txt ist vorhanden\n");
       }
 if(is_cgi_installed("php.cgi")){
       display("Das CGI php.cgi ist im Verzeichnis /cgi-bin
installiert\n");
       }
 if(!is_cgi_installed("/php.cgi")){
       display("Im Root-Verzeichnis des entfernten Web-Servers
existiert kein 'php.cgi' CGI-Script \n");
       }


#
# FTP
```

```
#
  # Verbindungsaufbau zum entfernten Host
 soc = open_sock_tcp(21);

 # Log in als anonymous user
 if(ftp_log_in(socket:soc, user:"ftp", pass:"joe@"))
 {
  # Erhalten eines passiven Ports
  port = ftp_get_pasv_port(socket:soc);
  if(port)
  {
   soc2 = open_sock_tcp(port);
   data = string("RETR /etc/passwd\r\n");
   send(socket:soc, data:data);
   password_file = recv(socket:soc2, length:10000);
   display(password_file);
   close(soc2);
  }
  close(soc);
 }
```

Mithilfe von NASL kann man IP-Pakete zusammenstellen und versuchen, diese effektiv einzusetzen. Ihre Prüfsumme wird automatisch neu berechnet, wenn Parameter des TCP-Pakets verändert wurden. Wird ein Layer an ein IP-Paket angehängt, wird das *ip_len*-Element des IP-Pakets aktualisiert.

Alle sogenannten Raw-Packet-Funktionen verwenden nicht-anonyme Argumente. Die Namen werden durch BSD Include-Dateien bestimmt. Aus diesem Grund wird das Element *length* eines IP-Pakets nicht mit *length*, sondern mit *ip_len* bezeichnet. Das Erstellen eines IP-Pakets ist recht einfach. Dazu verwendet man die Funktion *forge_ip_packet()*. Die Funktion *get_ip_element()* liefert Paketelemente, während die Funktion *set_ip_elements()* Elemente eines existierenden IP-Pakets verändern kann.

Ein Beispiel:

```
<return_value> = forge_ip_packet(
                            ip_hl    : <ip_hl>,
                            ip_v     : <ip_v>,
                            ip_tos   : <ip_tos>,
                            ip_len   : <ip_len>,
                            ip_id    : <ip_id>,
                            ip_off   : <ip_off>,
                            ip_ttl   : <ip_ttl>,
                            ip_p     : <ip_p>,
                            ip_src   : <ip_src>,
                            ip_dst   : <ip_dst>,
                            [ip_sum  : <ip_sum>] );
```

Das *ip_sum*-Argument dieser Funktion ist optional. Das Feld *ip_p* kann einen numerischen Wert oder eine der folgenden Konstanten beinhalten: *IPPROTO_TCP*, *IPPROTO_UDP*, *IPPROTO_ICMP*, *IPPROTO_IGMP* oder *IPPROTO_IP*.

Ein Beispiel:

```
<element> = get_ip_element(
             ip     : <ip_variable>,
             element:"ip_hl"|"ip_v"|"ip_tos"|"ip_len"|
          "ip_id"|"ip_off"|"ip_ttl"|"ip_p"|
          "ip_sum"|"ip_src"|"ip_dst");
```

Die Funktion *get_ip_element()* liefert ein bestimmtes Paket-Element. Elemente können sein: *ip_hl*, *ip_v*, *ip_tos*, *ip_len*, *ip_id*, *ip_off*, *ip_ttl*, *ip_p*, *ip_sum*, *ip_src* oder *ip_dst*.

Ein Beispiel:

```
set_ip_elements( ip : <ip_variable>,
                      [ip_hl    : <ip_hl>, ]
                      [ip_v     : <ip_v>,  ]
                      [ip_tos   : <ip_tos>,]
                      [ip_len   : <ip_len>,]
                      [ip_id    : <ip_id>, ]
                      [ip_off   : <ip_off>,]
                      [ip_ttl   : <ip_ttl>,]
                      [ip_p     : <ip_p>,  ]
                      [ip_src   : <ip_src>,]
                      [ip_dst   : <ip_dst>,]
                      [ip_sum  : <ip_sum>  ]
                    );
```

Die Funktion s*et_ip_elements()* verändert den Inhalt des IP-Pakets *<ip_variable>* und berechnet die Prüfsumme neu, wenn das Element *ip_sum* nicht explizit angegeben wurde. Diese Funktion erzeugt im Speicher kein neues Paket, deshalb sollte, wenn viele ähnliche IP-Pakete abzusetzen sind, *forge_ip_packet()* vorgezogen werden.

Schließlich verwendet man die Funktion *dump_ip_packet(<packet>)*, die das IP-Paket in lesbarer Form am Bildschirm ausgibt. Der Einsatz dieser Funktion macht nur bei der Fehlersuche Sinn. Um TCP-Pakete zu erzeugen, verwendet man die Funktion *forge_tcp_packet()*.

Ein Beispiel:

```
tcppacket = forge_tcp_packet(ip : <ip_packet>,
         th_sport : <source_port>,
                    th_dport : <destination_port>,
                    th_flags : <tcp_flags>,
                    th_seq   : <sequence_number>,
                    th_ack   : <acknowledgement_number>,
                    [th_x2   : <unused>],
                    th_off   : <offset>,
                    th_win   : <window>,
                    th_urp   : <urgent_pointer>,
                    [th_sum  : <checksum>],
                    [data    : <data>]);
```

Wichtig ist dabei, dass die Option *th_flags* aus *TH_SYN*, *TH_ACK*, *TH_FIN*, *TH_PUSH* oder *TH_RST* bestehen muss. Flags können unter Verwendung des |-Operators miteinander kombiniert werden. *th_flags* können aber auch numerischen Inhalt annehmen. Die Funktion *ip_packet* muss über *forge_ip_packet()* generiert werden oder aus einem über *send_packet()* oder *pcap_next()* gelesenen Paket bestehen.

Man verwendet die Funktion *set_tcp_elements()*, um TCP-Elemente zu verändern. Seine Syntax ist der *forge_tcp_packet()*-Syntax ähnlich.

Ein Beispiel:

```
set_tcp_elements(tcp : <tcp_packet>,
         [th_sport : <source_port>,]
                   [th_dport : <destination_port>,]
                   [th_flags : <tcp_flags>,]
                   [th_seq   : <sequence_number>,]
                   [th_ack   : <acknowledge-ment_number>,]
                   [th_x2    : <unused>,]
                   [th_off   : <offset>,]
                   [th_win   : <window>,]
                   [th_urp   : <urgent_pointer>,]
                   [th_sum   : <checksum>],
                   [data     : <data>] );
```

Um ein Element eines TCP-Pakets zu erhalten, verwendet man die Funktion *get_tcp_element()*. Die Syntax lautet:

```
element = get_tcp_elements(tcp: <tcp_packet>,
                  element: <element_name>);
```

Das Erstellen eines UDP-Pakets ist ähnlich einfach. Dazu bedient man sich der UDP-Funktion. Deren Verwendung ist der TCP-Funktion sehr ähnlich:

```
udp = forge_udp_packet(ip:<ip_packet>,
             uh_sport : <source_port>,
                        uh_dport : <destination_port>,
                        uh_ulen  : <length>,
                        [uh_sum  : <checksum>],
                        [data    : <data>]);
```

Die Funktionen *set_udp_elements()* und *get_udp_elements()* funktionieren wie die TCP-Funktionen.

9.3 NASL-Hilfsmittel

NASL verfügt über verschiedene Hilfsfunktionen, die helfen, verschiedene Aufgaben zu vereinfachen. Die Funktionen im Überblick:

- **this_host()**: Diese Funktion verwendet kein Argument und liefert die IP-Adresse des Hosts, auf dem das Skript ausgeführt wird.
- **get_host_name()**: Diese Funktion verwendet kein Argument und liefert den Namen des aktuell getesteten Host.
- **get_host_ip()**: Diese Funktion verwendet kein Argument und liefert die IP-Adresse des aktuell analysierten Hosts.
- **get_host_open_port()**: Diese Funktion gibt die Nummer des ersten offenen TCP-Ports am entfernten Rechner aus.
- **get_port_state(<portnum>)**: Diese Funktion gibt den Wert *TRUE* aus, wenn der TCP-Port <*port-num*> offen oder sein Zustand unbekannt ist.
- **telnet_init(<soc>)**: Diese Funktion erzeugt eine Telnet-Session über das geöffnete Socket <*soc*> und liefert die erste Zeile der Telnet-Daten. Hier ein Beispiel für den Einsatz der Hilfsfunktion:

```
soc = open_sock_tcp(23);

buffer = telnet_init(soc);

display("Der entfernte Telnet Banner ist: ", buffer,
"\n");
```

- **tcp_ping()**: Diese Funktion gibt den Wert TRUE aus, wenn der entfernte Host auf den TCP-Ping Request reagiert (TCP-Paket mit gesetztem ACK-Flag).
- **getrpcport()**: Diese Funktion entspricht der Standardfunktion desselben Namens. Die Syntax:

```
result = getrpcport(program : <pro-gram_number>,
       protocol: IPPROTO_TCP|IPPROTO_UDP,

       [version: <ver-sion>]);
```

9.4 Manipulation von Zeichenketten

NASL unterstützt auch die Manipulation von Zeichenketten. Dabei behandeln die Skripts Zeichenketten als Zahlen. Daher können auch die Operatoren ==, <, und > verwendet werden.

Ein Beispiel:

```
a = "version 1.2.3";
b = "version 1.4.1";

if(a < b){
#
# Wird durchgefuehrt, so lange version
# 1.2.3 kleiner ist als version 1.4.1
      }

c = "version 1.2.3";

if(a==c) {
      # wird ebenso überprüft
      }
```

Mit NASL kann man auch, wie in C, das n-te Zeichen einer Zeichenkette bestimmen:

```
a = "test";
b = a[1];  # b wird auf den Wert "e" gesetzt (= 0-relativ)
```

Zeichenketten können außerdem addiert und subtrahiert werden:

```
a = "version 1.2.3";
b = a - "version ";   # b erhält den Wert "1.2.3"

a = "Das ist ein Test";
b = " ist ein ";
c = a - b;            # c erhält den Wert "Das Test"

a = "test";
```

```
a = a+a;              # a erhält den Wert "testtest"
```

Für den Vergleich von Mustern verwendet man die Funktion *ereg()*. Die Syntax:

```
result = ereg(pattern:<pattern>, string:<string>)
```

Die *pattern*-Syntax entspricht dem *egrep*-Stil. Und so setzt man die Funktion ein:

```
if(ereg(pattern:".*", string:"test"))
{
  display("Wird immer durchgeführt\n");
}

mystring = recv(socket:soc, length:1024);
if(ereg(pattern: "SSH-.*-1\..*",
       string : mystring
       ))
{
 display("SSH 1.x is running on this host");
}
```

Will man Buffer-Overflow-Tests durchführen, so greift man zur Funktion *crap()*. Sie hat zwei Schreibweisen:

- **crap(<length>)**: Sie liefert eine Zeichenkette der Länge <*length*>.
- **crap(length:<length>, data:<data>)**: Sie liefert eine Zeichenkette der Länge <*length*>, die aus dem sich ständig wiederholenden *data*-Inhalt besteht.

Ein Beispiel:

```
a = crap(5);          # a = "XXXXX";
b = crap(4096);       # b = "XXXX...XXXX" (4096 X's)
c = crap(length:12,   # c = "hellohellohe" (length: 12);
         data:"hello");
```

Um Zeichenketten in anderen Zeichenketten zu verwenden, greift man zur *string()*-Funktion. Die Syntax lautet:

```
string(<string1>, [<string2>, ..., <stringN>])
```

Wichtig ist dabei, dass die Funktion Steuerzeichen wie \n oder \t interpoliert. Ein Beispiel:

```
name = "herbert";

a = string("Hallo, Ich bin ", name, "\n");    # a wird mit
Inhalt "Hello, Ich bin herbert" #belegt
# (mit Steuerzeichen NewLine am Ende)

b = string(1, " und ", 2, " macht ", 1+2);
# b wird mit Inhalt  "1 und 2 macht 3" belegt

c = string("MKD ", crap(4096), "\r\n");
# c wird mit Inhalt "MKD XXXXX.....XXXX"
# belegt
# (4096 X) gefolgt von einem CarriageReturn
#  und einem NewLine
```

Die Funktion *strlen()* liefert die Länge einer Zeichenkette. Ein Beispiel:

```
a = strlen("abcd"); # a erhält den Wert 4
```

Die Funktion *strtoint()* konvertiert einen NASL-Integerwert in einen Integerwert binärer Basis. Die Syntax ist:

```
value = strtoint(number:<nasl_integer>, si-
ze:<number_of_bytes>);
```

Diese Funktion ist in Verbindung mit *raw_string()* anwendbar. Das *size*-Argument entspricht der Byte-Anzahl, in welches der NASL-Integer konvertiert werden soll, und kann den Wert 1, 2 oder 4 annehmen. Schließlich kann die Funktion *tolower()*

verwendet werden, um eine gesamte Zeichenkette in Kleinbuchstaben zu konvertieren. Die Syntax:

```
tolower(<string>);
```

Diese Funktion liefert als Ergebnis die Zeichenkette <string> in Kleinbuchstaben. Ein Beispiel:

```
a = "Hallo mein Name ist herbert";
b = tolower(a);
# b erhält den Wert
# "hallo mein name ist herbert"
```

9.5 Eigene Plug-ins erstellen

Nach all der Theorie kommen wir jetzt zu einer der spannendsten Fragen: Wie schreibt man eigentlich eigene Testskripts? Bevor Sie sich an das Entwickeln eigener Skripts machen, sollten Sie wissen, dass *openvassd* die Tests der Reihe nach ausführt und dabei zwischen der Ausführung zweier Skripts eine kurze Pause einlegt.

Für die effektive Skriptentwicklung bedeutet das, dass Tests auf den Ergebnissen anderer Sicherheitstests aufbauen sollten. So sollte beispielsweise vor der Überprüfung eines bestimmten Dienstes getestet werden, ob der Standardport für diesen Dienst verfügbar ist. Ein derart logischer Aufbau führt zu schnelleren und auch besseren Ergebnissen.

Von besonderer Bedeutung ist daher die Funktion *get_port_state(<portnum>)*, welche die offenen Ports bestimmt. Sie liefert den Ausgabewert *TRUE*, wenn der Port offen ist, ansonsten den Wert *FALSE*. Diese Funktion liefert auch den Wert *TRUE*, wenn der Port nicht gescannt wurde oder sein Status (Zustand) unbekannt ist. Ein weiterer Vorteil dieser Funktion ist, dass sie wenig Rechenleistung verlangt.

Eine weitere wichtige OpenVAS-Funktion ist die Knowledge Base (KB). In ihr werden sämtliche Scan-Informationen über die einzelnen Sicherheitstests gespeichert. Und die Sicherheitstests greifen auf diese Informationen zu, um beispielsweise Informationen über Port-Stati abzufragen. Die Knowledge Base ist in Kategorien unterteilt. Die Kategorie *Services* beinhaltet die Port-Nummern mit den bekannten Services. So enthält beispielsweise das Element *Services/smtp* den Wert *25*. Zwei Funktionen beziehen sich auf die Knowledge Base:

- **get_kb_item(<name>)**: Diese Funktion liefert den Inhalt des Knowledge Base-Feldes <name>. Diese Funktion ist anonym.
- **set_kb_item(name:<name>, value:<value>)**: Diese Funktion setzt das Feld <*name*> auf den Wert <value>.

Widmen wir uns dem Aufbau von NASL-Skripts. Jedes NASL-Skript registriert sich selbstständig am OpenVAS-Server. Es übergibt *openvassd* seinen Namen, seine Beschreibung, den Namen des Autors und weitere Informationen. Ein NASL-Skript besteht aus zwei Abschnitten:

- **Registriersektion**: Hier sind allgemeine Informationen, wie Autor, Beschreibung etc., über das Skript hinterlegt.
- **Attack-Sektion**: Hier sind die eigentlichen Befehle für die Testdurchführung zu finden.

Generell besitzt ein NASL-Skript folgende Struktur:

```
#
# NASL-Skript verwendet von openvassd
#


if(description)
{
# Registrierinformationen
# „Registriersektion“
#

 exit(0);
}

#
# Skript-Code  - „Attack-Sektion“
#
```

Eine Besonderheit ist die globale Variable *description*. Abhängig davon, ob ein Skript sich registrieren muss oder nicht, wird sie auf *TRUE* oder *FALSE* gesetzt.

In der Registriersektion sind folgende Funktionen aufzurufen:

- **script_name(language1:<name>, [...])**: Diese Funktion legt die Skriptbezeichnung fest, wie sie im Client erscheinen soll.
- **script_description(language1:<desc>, [...])**: Diese Funktion legt die Skriptbeschreibung fest, wie sie im Client erscheinen soll, wenn der User den Namen anklickt.
- **script_summary(language1:<summary>, [...])**: Diese Funktion dient der Skriptbeschreibung, wie sie in den Tooltipps erscheinen soll. Die Beschreibung darf nicht länger als eine Zeile sein.
- **script_category(<category>)**: Diese Funktion legt die Skriptkategorie fest. Es muss sich um eine der folgenden Kategorien handeln:
    - ACT_GATHER_INFO: Diese Skripts werden vorab durchgeführt und können auf keinen Fall den Zielhost schädigen.
    - ACT_ATTACK: Sie versuchen, Rechte am entfernten Rechner zu erhalten, und sind in der Lage, den entfernten Rechner zu schädigen (beispielsweise ein Buffer-Overflow-Test).
    - ACT_DENIAL: Sie versuchen, den entfernten Rechner zum Absturz zu bringen.
    - ACT_SCANNER: Sie führen typische Port-Scans durch.

- **script_copyright(language1:<copyright>, [...]**: Mit dieser Funktion hinterlegt man das Copyright des Skripts.
- **script_family(language1:<family>, [...])**: Diese Funktion legt die Skript-Familie fest. Es stehen folgende Familien zur Auswahl:
    - Backdoors
    - CGI abuses
    - Denial of Service
    - FTP
    - Finger abuses
    - Firewalls
    - Gain a shell remotely
    - Gain root remotely
    - Misc.
    - NIS
    - RPC
    - Remote file access
    - SMTP problems
    - Useless services

NASL bietet auch Multilingual-Support. Jedes Skript muss mindestens Englisch unterstützen. Die Syntax lautet:

```
script_Funktion(english:englischer_text,
    [francais:französischer_text,
     deutsch:deutscher_text,
     ]);
```

Außerdem kann in der Registrier-Sektion die Funktion *script_dependencies()* aufgerufen werden. Mit ihr lassen sich Skript-Abhängigkeiten definieren. Diese Funktion teilt *openvassd* mit, dass das aktuelle Skript erst nach einem anderen durchzuführen ist. Die Syntax für die Abhängigkeiten:

```
script_dependencies(filename1 [,filename2, ..., filenameN]);
```

Dabei gibt *filename* die Bezeichnung des Skripts an, das später in Abhängigkeit ausgeführt werden soll. Dabei können entsprechend obiger Syntax auch mehrere Skripts angegeben werden. In der Attack-Sektion definiert man die eigentlichen Tests und all das, was für die Durchführung wichtig ist. Außerdem können Probleminformationen für ähnlich arbeitende Tests, unter Verwendung der *security_warning()* und *security_hole()* Funktionen abgelegt werden. Die Funktion *security_warning()* kommt zum Einsatz, wenn eine Attacke zwar erfolgreich war, aber kein großes Sicherheitsproblem darstellt. Diese beiden Funktionen haben folgende Syntax:

```
security_warning(<port> [, proto-col:<proto>]);
security_hole(<port> [, protocol:<proto>]);

security_warning(port:<port>, data:<data> [, proto-
col:<proto>]);
security_hole(port:<port>, data:<data> [, protocol:<proto>]);
```

Im vorangegangenen Beispiel wird die Skriptbeschreibung am Client angezeigt, wie in der *script_description()* angegeben. Das ist aufgrund des Multilingualsupports sehr praktisch. Im zweiten Fall wird am Client der Inhalt des Arguments *data* angezeigt. Das ist praktisch, falls man Anzeigeinformationen, wie die Versionsnummer, auf schnellem Weg zur Anzeige bringen will.

Zusätzlich zum Sicherheitstest kann NASL auch für einige Unterstützungstätigkeiten genutzt werden. Nachfolgendes Beispiel prüft, ob auf allen Hosts SSH läuft. Es erfolgt eine Mitteilung, wenn ein Host ohne SSH gefunden wird:

```
#
# Überprüfe ssh
#
if(description)
{
 script_name(english:"Ensure the presence of ssh");
 script_description(english:"This script makes sure that ssh
is running");
 script_summary(english:"connects on remote tcp port 22");
 script_category(ACT_GATHER_INFO);
 script_family(english:"Administration tool-box");
 script_copyright(english:"This script was written by Joe
U.");
 script_dependencies("find_service.nes");
 exit(0);
}

#
# First, ssh may run on another port.
# That's why we rely on the plugin 'find_service'
#

port = get_kb_item("Services/ssh");
if(!port)port = 22;
# declare that ssh is not installed yet
ok = 0;
if(get_port_state(port))
{
 soc = open_sock_tcp(port);
```

```
 if(soc)
 {
  # Check that ssh is not tcpwrapped. And that it's really
  # SSH
  data = recv(socket:soc, length:200);
  if("SSH" >< data)ok = 1;
 }
 close(soc);
}

#
# Only warn the user that SSH is NOT installed
#
if(!ok)
{
  report = "SSH is not running on this host !";
  security_warning(port:22, data:report);
}
```

9.6 Feinschliff

OpenVAS kann bei seinen Sicherheitstests auf unzählige Standardskripts zurückgreifen. Um eine möglichst flotte Durchführung sicherstellen zu können, sollte man seine eigenen Skripts optimieren. Der einfachste Weg, um ein Skript zu optimieren: Man teilt *openvassd* mit, in welchem Fall das Skript nicht ausgeführt werden soll. Angenommen, ein Skript versucht, eine Verbindung zu einem Port x aufzubauen.

Liegen dem Scanner-Prozess bereits Informationen über diesen Port vor, so macht es natürlich keinen Sinn, diesen erneut zu testen. Für solche Fälle wurden die Funktionen *script_require_ports()*, *script_require_keys()* und *script_ex-clu-de _keys()* geschaffen. Sie werden in der Beschreibung aufgerufen.

Die Funktionen im Einzelnen:

- **script_require_ports(<port1>, <port2>, ...)**: Diese Funktion sorgt dafür, dass *openvassd* die Durchführung des Skripts nur dann startet, wenn der Port tatsächlich offen ist. Ein Beispiel:
  ```
  script_require_ports(80, "Servi-ces/www")
  ```

  Ist der Status des Ports unbekannt, wird das Skript ausgeführt.
- **script_require_keys(<key1>, <key2>, ...)**: Diese Funktion führt das Skript nur dann aus, wenn alle Argumente in der Knowledge Base angegeben wurden.
- **script_exclude_keys(<key1>, <key2>, ...)**: Es wird kein Test durchgeführt, wenn eines der Schlüsselargumente in der Knowledge Base gesetzt ist.

Die Resultate anderer Skripts können aus der Knowledge Base bezogen werden.

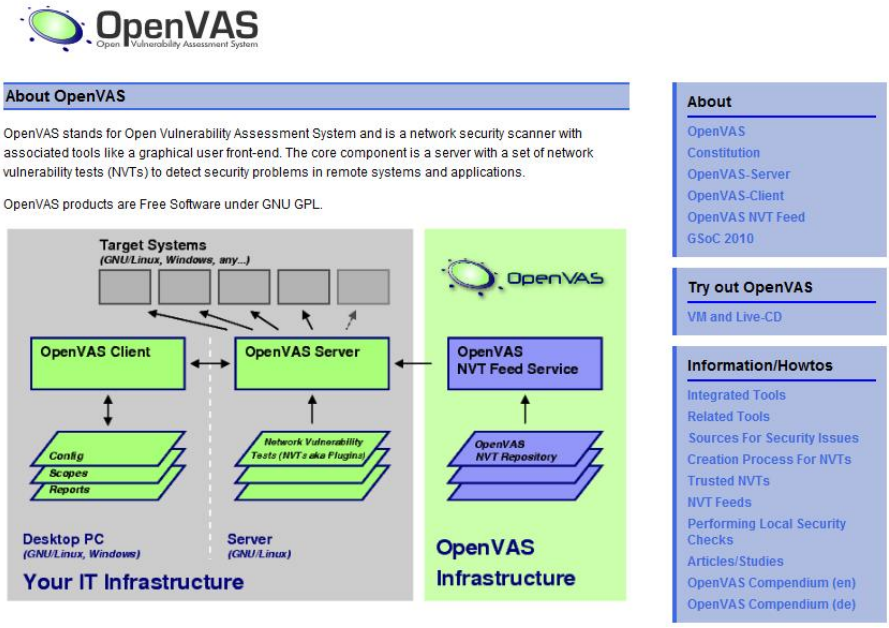
9.7 Skript-Weitergabe

Will man Eigenentwicklungen anderen Anwendern zur Verfügung stellen, so sollten folgende Regeln beachtet werden:

- Das Skript darf unter keinen Umständen mit einem User interagieren.
- NASL-Skripts werden am Server durchgeführt. Deshalb werden dem User Outputs nicht angezeigt. Ein Skript darf immer nur eine Schwachstelle testen.
- Für Multi-Verwundbarkeiten sollten separate Skripts erstellt werden, und es ist auf eine eindeutige Beschreibung und Kategorisierung zu achten.

Anhang A – More Info

Wie wir gesehen haben, ist OpenVAS ein ausgesprochen flexibler Scanner, der sich inzwischen nicht mehr nur auf das klassische Netzwerk-Scannen beschränkt, sondern auch die Untersuchung der Host-Installationen erlaubt. Wenn Sie den Scanner regelmäßig einsetzen wollen, werden Sie immer wieder auf Probleme treffen, neue Plug-ins oder einfach nur Rat von anderen Anwendern suchen.



Die wichtigste Anlaufstelle für OpenVAS-Anwender ist natürlich die Projekt-Homepage (*http://www.openvas.org*).

Die OpenVAS-Community stellt inzwischen ein umfangreiches Informationsangebot bereit. Erste Anlaufstelle ist natürlich die OpenVAS-Homepage (*http://www.openvas.org*), auf der Sie insbesondere die neuesten OpenVAS-Pakete finden.

Weitere wichtige Informationsquellen sind Mailinglisten und der Bugtracker. Eine Plug-in-Datenbank befindet sich im Aufbau. Auch für Entwickler gibt es eine ganze Menge Informationen. Ihrer aktiven Beteiligung an dem Projekt steht also nichts im Wege :-).

Wenn Sie professionelle Unterstützung beim Umgang mit OpenVAS benötigen, finden Sie insbesondere in der Intevation GmbH (http://intevation.de), die maßgeblich für die Entwicklung des Scanners verantwortlich ist, den richtigen Partner. Unter *http://www.openvas.org/professional-services.html* finden Sie eine Übersicht weiterer Dienstleister.

Anhang B – die OpenVAS-Dateien

Wenn Sie intensiv mit OpenVAS arbeiten wollen, so ist es immer sehr hilfreich zu wissen, welche Funktionalitäten der Umgebung durch welche Dateien realisiert werden. Dieser Anhang gibt Ihnen einen großen Überblick zu den wichtigsten OpenVAS-Dateien.

Beachten Sie bei den Pfadangaben lediglich, dass die hier beschriebenen Speicherorte sich auf eine Standardinstallation aus dem Quelltext mit *PREFIX=/usr/local* beziehen.

Ausführbare Dateien für Benutzer

Im Verzeichnis *PREFIX/bin* finden Sie die ausführbaren Dateien, die für den Benutzer oder während des Kompilierens nützlich sein können.

- **libopenvas-config**: Dient der Ausgabe von Kompilierungsparametern für Module, die *openvas-libraries* verwenden.
- **openvas-libnasl-config**: Übernimmt die Ausgabe von Kompilierungsparametern für Module, die *openvaslibnasl* verwenden.
- **openvasd-config**: Ist für die Ausgabe von Kompilierungsparametern für Module zuständig, die *openvas-server* verwenden.
- **openvas-mkcert-client**: Dient dem Erstellen von Client-Zertifikaten.
- **openvas-nasl**: Ein eigenständiger NASL-Interpreter.
- **OpenVAS-Client**: Das ist die grafische Benutzungsoberfläche des Clients.

Server-Konfiguration

Die Dateien für die Server-Konfiguration finden Sie im Verzeichnis *PREFIX/etc/openvas*. Neben den systemweiten Konfigurationsdateien für OpenVAS finden Sie dort auch die Zugriffsregeln und die Benutzerdatenbank. Die Dateien:

- **openvasd.conf**: Die zentrale Konfigurationsdatei des OpenVAS Servers.
- **openvasd.rules**: Zugriffsregeln, die den Zugriff auf Zielsysteme für die Nutzer dieser OpenVAS-Installation einschränken.
- **gnupg**: Der Keyring und die Vertrauensgrade, die bestimmen, welche NVTs vertrauenswürdig genug sind, damit sie ausgeführt werden können.

Datei für die Kompilierung

Dieses Verzeichnis *PREFIX/include/openvas* enthält die C-Header-Dateien für OpenVAS. Diese Dateien sind nur für die Kompilierung von OpenVAS auf dem jeweiligen System relevant.

Bibliotheken

Im Verzeichnis *PREFIX/lib* sind die statischen (.a) und die dynamischen (.so) Programmbibliotheken zu finden, die von den Modulen *openvas-libraries* und *openvas-libnasl* genutzt werden.

NVTs

Im Verzeichnis *PREFIX/lib/openvas/plugins* finden Sie über 17.000 nasl- und inc-Dateien. Dahinter verbergen sich die Testskripts.

Ausführbare Dateien des OpenVAS-Servers

Das Verzeichnis *PREFIX/sbin* enthält verschiedene ausführbare Dateien, die für die Administration von OpenVAS-Server relevant sind:

- **openvassd**: OpenVAS-Server.
- **openvas-adduser**: Programm zum Hinzufügen von OpenVAS-Nutzern.
- **openvas-mkcert**: Programm zum Erstellen eines Server-Zertifikats.
- **openvas-rmuser**: Programm zum Entfernen von OpenVAS-Nutzern.
- **openvas-nvt-sync**: Programm zur Synchronisation mit NVT Feeds.

Bedienungsanleitungen

Im Verzeichnis *PREFIX/share/man* finden Sie die Bedienungsanleitungen für die für OpenVAS-Anwender gedachten ausführbaren Dateien. Im Verzeichnis *PREFIX/man* sind die Bedienungsanleitungen für OpenVAS-Administratoren zu finden.

Installationsspezifische Daten

In dem Verzeichnis *PREFIX/var/lib/openvas* sind die installationsspezifischen Daten einer OpenVAS-Installation zu finden:

- **CA**: Öffentliche Zertifikate, die für den Aufbau einer verschlüsselten Verbindung erstellt wurden.
- **private/CA**: Die dazugehörigen geheimen Zertifikate.
- **users**: Dateien für die OpenVAS-Benutzer.
- **openvas-services**: Liste der bekannten Dienste und der zugehörigen Portnummern.
- **services.tcp**: TCP-Dienste
- **services.udp**: UDP-Dienste

Protokolldateien

Die Protokolldateien Ihrer OpenVAS-Installation finden Sie im Verzeichnis *PREFIX/var/log/openvas*. Der OpenVAS-Server legt zwei Protokolldateien an:

- *openvasd.dump*
- *openvasd.messages*

Serverprozessinformation

Wenn Sie sich für die Laufzeitdaten einer in der Ausführung befindlichen OpenVAS-Installation interessieren, werfen Sie einen Blick ins Verzeichnis *PREFIX/var/run*. Die Datei *openvasd.pid* dient der Prozessidentifizierung.

Benutzerdaten

Die benutzerspezifischen Daten, die vom OpenVAS-Client verwaltet werden, finden Sie im jeweiligen Benutzerverzeichnis */HOME*.

- **.openvas/A/**: Verzeichnis mit allen Dateien bezüglich Aufgabe A.
- **.openvas/A/openvasrc**: Aufgabenspezifische Konfiguration.
- **.openvas/A/B/**: Verzeichnis mit allen Dateien, die sich auf Bereich B der Aufgabe A beziehen.
- **.openvas/A/B/openvasrc**: Bereichsspezifische Konfiguration.
- **.openvas/A/B/C/**: Verzeichnis mit allen Dateien, die sich auf den Bericht C im Bereich B der Aufgabe A beziehen.
- **.openvas/A/B/C/openvasrc**: Die während der Erstellung des Berichts C gesetzte Konfiguration.
- **.openvas/A/B/C/report.nbe**: Der Inhalt des Berichts C im Textformat.
- **.openvas/A/B/C/report.nbe.cnt**: Zähler für die im jeweiligen Bericht enthaltenen Sicherheitsmeldungen. Dieser Zähler kann aus der Berichtsdatei *report.nbe* neu berechnet werden.
- **.openvas/A/B/RRR/openvas_nvt_cache**: Zwischenspeicher der Beschreibungen der NVTs, die bei der Erstellung des Berichts C eingesetzt wurden.
- **.openvasrc**: Voreinstellungen für OpenVAS-Client.
- **.openvasrc.cert**: Zertifikate von OpenVAS-Servern, mit denen sich der Client in der Vergangenheit verbunden hat und die vom Benutzer als vertrauenswürdig akzeptiert wurden.

Anhang C – das OpenVAS Management Protocol

In der aktuellen OpenVAS-Version kommunizieren der OpenVAS-Server und der Client noch per OpenVAS Transfer Protocol, kurz OTP, miteinander. Dieses Protokoll wird durch das neue OpenVAS Management Protocol, kurz OMP, abgelöst. Das neue Protokoll wird einige interessante Funktionen ermöglichen. So wird es beispielsweise ein konsistentes, privilegientrennendes Server-Management von Benutzerdaten wie Aufgaben, Berichte etc., erlauben. Auch für Entwickler wird sich durch die Einführung die Entwicklung von HTTP- bzw. XML-basierten Implementierungen vereinfachen.

OMP im Überblick

OTP ist ein Abkömmling von Nessus Transfer Protocol (NTP), das allerdings schon um einige nicht benötigte, unsichere und inkonsistente Elemente bereinigt wurde. Da OTP aber immer noch einige Schwächen seines Vorgänger mit sich trägt, hat sich die OpenVAS-Community auf die Einführung eines neuen Kommunikationsprotokolls geeignet, das in einer „sauberen und modernen Art“ daherkommt, aber auch offen für zukünftige Verbesserungen ist.

Die wichtigen Ziele des neuen Protokolls:

- Er soll einfach und schnell zu parsen sein, gleichzeitig für jedermann leicht verständlich sein, um das Debugging und die Entwicklung zu vereinfachen.
- Es soll so statuslos sein wie nur irgend möglich, um unnötige Verzögerungen bei der Kommunikation zwischen Client und Server zu verhindern.
- Es soll sich einfach in andere Protokolle und Kommunikationskanäle integrieren.

Diesen Anforderungen schien ein XML-basierten Protokoll am besten zu genügen.

Durch die Einführung von OMP wird die OpenVAS-Architektur allerdings auch deutlich komplexer, weil eine neue Ebene zwischen dem OpenVAS-Server und

dem Client einführt wird. Diese trägt die Bezeichnung OpenVAS-Manager, verwendet OMP für die Kommunikation mit den Clients, speichert aber Benutzerdaten und kommuniziert weiter per OTP mit dem Server. Zukünftige OpenVAS-Clients werden also OMP und OTP unterstützen.

OMP stellt folgende Grundfunktionen zur Verfügung:

- abort_task – beendet eine Aufgabe
- create_task – erzeugt eine neue Aufgabe
- delete_report – löscht einen existierenden Bericht
- delete_task – löscht eine Aufgabe
- get_certificates – holt sich alle verfügbaren Zertifikate
- get_dependencies – holt sich die Abhängigkeiten aller verfügbaren NVTs
- get_nvt_all – holt sich die IDs und Namen aller verfügbaren NVTs
- get_nvt_details – holt sich die Details aller verfügbaren NVTs
- get_nvt_feed_checksum – holt sich die Prüfsumme der gesamten NVT-Sammlung
- get_preferences – holt die Voreinstellung aller verfügbaren NVTs
- get_report – holt sich den Bericht einer bestimmten ID
- get_rules – holt sich die Regeln für einen authentifizierten Benutzer
- get_status – holt sich die Aufgabenstatusinformationen
- get_version – holt sich die OpenVAS Manager-Protokollversion
- help – gibt die Hilfe aus
- modify_report – modifiziert einen bestehenden Bericht
- modify_task – aktualisiert eine bestehende Aufgabe
- start_task – startet eine existierende Aufgabe

Erstellen einer Aufgabe

Exemplarisch schauen wir uns einige gängige Aktionen an. Für das Erstellen einer Aufgabe ist der Befehl *create_task* zuständig.

In den Befehl muss eine Datei eingeschlossen werden, die die Aufgabe spezifiziert. Auch Kommentare können angegeben werden.

Es gibt folgende Kommando-Elemente:

- <**rcfile**>: Datei mit der Aufgabenbeschreibung. Sie besitzt das openvasrc-Format.
- <**name**>: Lesbare ID mit einer Länge von maximal 100 Zeichen.
- <**comment**>: Optionaler Kommentar mit maximal 4000 Zeichen.

Hier ein Beispiel für die Kommunikation zwischen dem Client und dem OpenVAS-Manager

C:

```
<create_task>
  <rcfile>asdf3235saf3kjBVF...</rcfile>
  <name>Scan Webserver</name>
  <comment>Hourly scan of the webserver</comment>
</create_task>
```

M:

```
<create_task_response status="201" id="254cd3ef-bbe1-4d58-
859d-21b8d0c046c6" />
```

M:

```
<create_task_response status="4xx" id="254cd3ef-bbe1-4d58-
859d-21b8d0c046c6" />
```

M:

```
<create_task_response status="5xx" id="254cd3ef-bbe1-4d58-
859d-21b8d0c046c6" />
```

Aufgabe starten

Das Starten einer Aufgabe wird mit dem Kommando *start_task* ausgeführt. Dabei muss die ID einer bestehenden Aufgabe angegeben werden.

Auch hierzu ein kleines Beispiel:

C:

```
<start_task task_id="825a5d10-24b2-4473-a4e0-55f8cfd4bf23" />
```

M:

```
<start_task_response status="201" task_id="825a5d10-24b2-
4473-a4e0-55f8cfd4bf23" />
```

M:

```
<start_task_response status="4xx" task_id="825a5d10-24b2-
4473-a4e0-55f8cfd4bf23" />
```

M:

```
<start_task_response status="5xx" task_id="825a5d10-24b2-
4473-a4e0-55f8cfd4bf23" />
```

Status abfragen

Der Client verwendet den Befehl *get_status*, um den Status einer Aufgabe abzufragen. Wird das Kommando ohne die Angabe einer Aufgaben-ID ausgeführt, so gibt der Manager eine Liste der Aufgaben zurück, die von dem Benutzer ausgeführt werden.

Wird mit dem Kommando eine ID übermittelt, gibt der Manager die Details an den Client zurück.

Auch dazu ein Beispiel:

C:

```
<get_status />
```

M:

```
<get_status_response status="200">
  <task_count>2</task_count>
  <task id="254cd3ef-bbe1-4d58-859d-21b8d0c046c6">
    <name>Scan Webserver</name>
    <status>Running</status>
    <messages>
      <hole>0</hole>
      <warning>0</warning>
      <info>0</info>
      <log>0</log>
      <debug>0</debug>
    </messages>
  </task>
  <task id="f14747d3-a4d7-4e79-99bb-a0a1276cb78c">
    <name>Scan Mailserver</name>
    <status>Done</status>
    <messages>
      <hole>0</hole>
      <warning>2</warning>
      <info>5</info>
      <log>0</log>
      <debug>7</debug>
    </messages>
  </task>
</get_status_response>
```

C:

```
<get_status task_id="f14747d3-a4d7-4e79-99bb-a0a1276cb78c" />
```

M:

```
<get_status_response status="200">
  <name>Scan Mailserver</name>
  <status>Done</status>
  <report_count>5</report_count>
  <report id="fc2ae4a9-8819-4159-b94b-5210db2f6f38">
    <timestamp>2009-03-10T10:21Z</timestamp>
    <messages>
      <hole>1</hole>
      <warning>2</warning>
      <info>5</info>
      <log>3</log>
      <debug>3</debug>
    </messages>
    <comment/>
  </report>
  <report id="bcfeca57-0068-494b-a0bc-7b056649bd8b">
    <timestamp>2009-03-10T10:29Z</timestamp>
    <messages>
      <hole>0</hole>
      <warning>1</warning>
      <info>5</info>
      <log>2</log>
      <debug>3</debug>
    </messages>
    <comment>Applied patch</comment>
  </report>
  <report id="0c176f64-faf8-4544-8de7-12dbccb3831c">
    <timestamp>2009-03-10T11:19Z</timestamp>
    <messages>
```

```
      <hole>0</hole>
      <warning>0</warning>
      <info>3</info>
      <log>2</log>
      <debug>3</debug>
    </messages>
    <comment/>
  </report>
  <report id="80d0096d-4216-42a7-b727-e6955c08f97b">
    <timestamp>2009-03-11T16:42Z</timestamp>
    <messages>
      <hole>0</hole>
      <warning>0</warning>
      <info>2</info>
      <log>2</log>
      <debug>3</debug>
    </messages>
    <comment/>
  </report>
  <report id="5d6294dd-e634-4f06-ae64-ef97e68b0e43">
    <timestamp>2009-03-11T17:02Z</timestamp>
    <messages>
      <hole>0</hole>
      <warning>0</warning>
      <info>1</info>
      <log>1</log>
      <debug>3</debug>
    </messages>
    <comment/>
  </report>
```

```
</get_status_response>
```

C:

```
<get_status task_id="254cd3ef-bbe1-4d58-859d-21b8d0c046c6" />
```

M:

```
<get_status_response status="200">
  <task id="254cd3ef-bbe1-4d58-859d-21b8d0c046c6">
    <name>Scan Mailserver</name>
    <status>Running</status>
    <current_ip>192.168.1.5</current_ip>
    <messages>
      <hole>0</hole>
      <warning>0</warning>
      <info>1</info>
      <log>1</log>
      <debug>3</debug>
    </messages>
  </task>
</get_status_response>
```

OIDs und NVT-Namen anfordern

Mit dem Befehl *get_nvtg_all* fordert der Client die OIDs und Namen der gesamten NVT-Sammlung an, die beim Manager verfügbar ist.

Ein Beispiel:

C:

```
<get_nvt_all/>
```

M:

```
<get_nvt_all_response status="200">
  <nvt_count>2432</nvt_count>
  <feed_checksum algo-
rithm="md5">931db829a06b9a440a2ecaeb68c03715</feed_checksum>
  <nvt oid="1.3.6.1.4.1.25623.1.7.13005">
    <name>FooBar 1.5 installed</name>
    <checksum
algorithm="md5">2ebef00ea4617c096849fb708864b0e7</checksum>
  </nvt>
  <nvt oid="1.3.6.1.4.1.25623.1.7.13006">
    <name>FooBar 2.1 XSS vulnerability</name>
    <checksum algo-
rithm="md5">ff4c6c4eab15bfc86bbb56f77b787929</checksum>
  </nvt>
  ...
  <nvt oid="1.3.6.1.4.1.25623.1.7.16021">
    <name>XYZ 1.1 vulnerability</name>
    <checksum algo-
rithm="md5">f4a09716c3b3df391072c77f42dff363</checksum>
  </nvt>
</get_nvt_all_response>
```

Einstellungen

Der Client holt sich mit dem Befehl *get_preferences* die Voreinstellungen für die NVT, die über den Manager verfügbar sind. Ein Beispiel für die typische Kommunikation zwischen OpenVAS-Client und -Manager:

C:

```
<get_preferences/>
```

M:

```
<get_preferences_response status="200">
  // TODO
</get_preferences_response>
```

C:

```
<get_preferences oid="1.3.6.1.4.1.25623.1.7.13005" />
```

M:

```
<get_preferences_response status="200">
  // TODO
</get_preferences_response>
```

Regeln abfragen

Mit dem Kommando *get_rules* fordert der Client die Regeln an. Der Manager reagiert mit einem Statuscode, der den Erfolg anzeigt, und den Regeln, die für den Benutzer gelten.

Auch hierzu ein Beispiel:

C:

```
<get_rules/>
```

M:

```
<get_rules_response status="200">
  // TODO
</get_rules_response>
```

Weitere Hilfe

Weitere Hilfe rufen Sie mit dem *help*-Befehl ab. Auch hierzu ein letztes Beispiel:

C:

```
<help/>
```

M:

```
<help_response status="200">
  ABORT_TASK          Abort a running task.
  AUTHENTICATE        Authenticate with the manager.
  CREATE_TASK         Create a new task.
  DELETE_REPORT       Delete an existing report.
  DELETE_TASK         Delete an existing task.
  GET_CERTIFICATES    Get all available certificates.
  GET_DEPENDENCIES    Get dependencies for all available
NVTs.
  GET_NVT_ALL         Get IDs and names of all available
NVTs.
  GET_NVT_FEED_CHECKSUM  Get checksum for entire NVT collec-
tion.
  GET_NVT_DETAILS     Get all details for all available NVTs.
  GET_PREFERENCES     Get preferences for all available NVTs.
  GET_REPORT          Get a report identified by its unique
ID.
  GET_RULES           Get the rules for the authenticated
user.
  GET_STATUS          Get task status information.
```

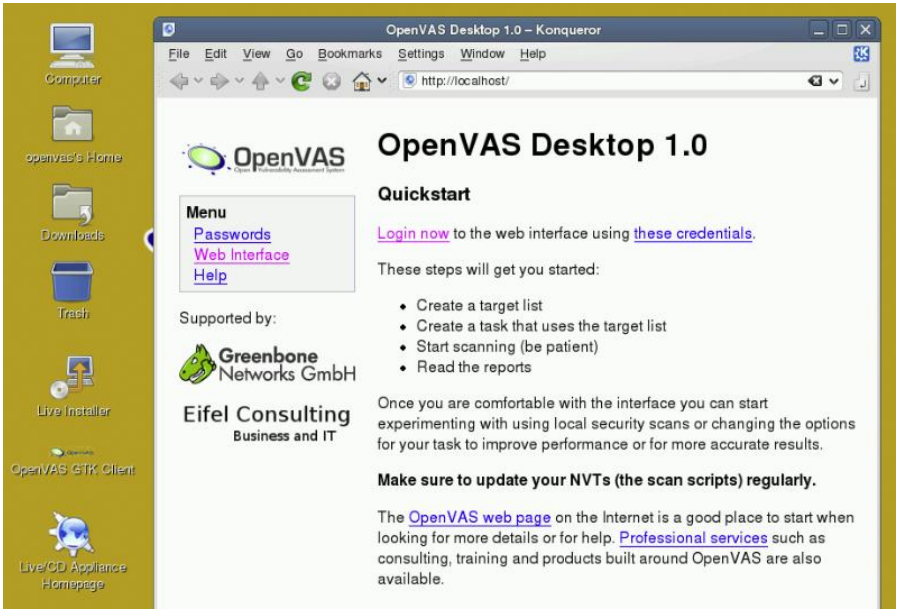
```
  GET_VERSION         Get the OpenVAS Manager Protocol ver-
sion.
  HELP                Get this help text.
  MODIFY_REPORT       Modify an existing report.
  MODIFY_TASK         Update an existing task.
  START_TASK          Manually start an existing task.
</help_response>
```

Wenn Sie tiefer in die OMP-Materie einsteigen wollen, finden Sie auf der OpenVAS-Website unter *http://www.openvas.org/openvas-cr-28.html* weitere Informationen.

Anhang D – die beiliegende CD

Auf der beiliegenden CD finden Sie eine Live-CD, mit der Sie OpenVAS mit all seinen Komponenten direkt testen können. Einfacher und mit weniger Aufwand können Sie OpenVAS und seine Funktionalität nicht evaluieren.

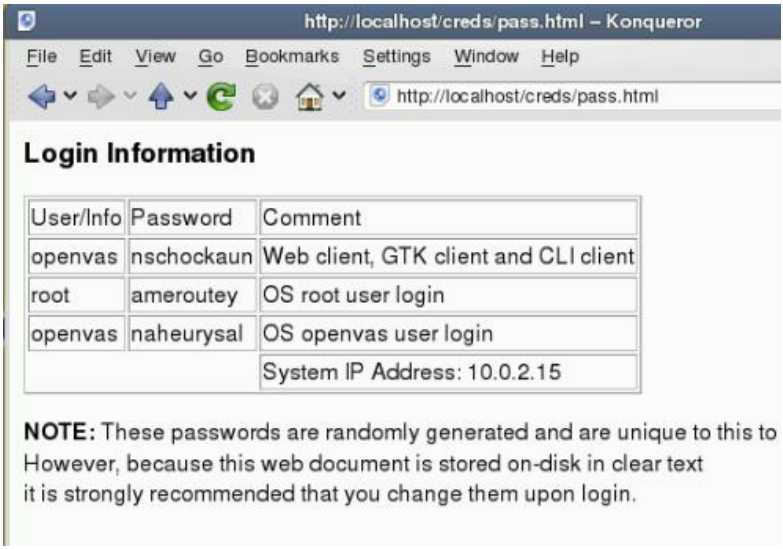
Die Nutzung der CD ist einfach: Legen Sie diese einfach in ein bootfähiges CD-ROM-Laufwerk und starten Sie den Rechner. Folgen Sie den Schritten am Bildschirm und nehmen Sie gegebenenfalls die notwendigen Änderungen (Sprachvariante, Host- und Domainname, Netzwerkschnittstelle) vor.



Das Live-System OpenVAS Desktop 1.0 erlaubt den bequemen Zugriff auf das OpenVAS-System.

Nach dem Start der Live-CD meldet sich ein webbasierter Start-Dialog, der Ihnen den Zugriff per GSA auf das OpenVAS-System erlaubt. Die Benutzerkennung und

das Passwort für den GSA, den OpenVAS-Clients und Root finden Sie auf der Seite *Passwords*. Diese Daten werden dynamisch beim Boot-Vorgang erzeugt, sind also immer unterschiedlich. Unter *http://localhost/creds/pass.html* erfahren Sie auch, unter welcher IP-Adresse das System zu erreichen ist.



Die dynamisch erstellten Zugangsdaten.

Mit der Zugangskennung *openvas* und dem auf der Password-Seite angezeigten Passwort können Sie über den *Login now*-Link den GSA nutzen.

Auf dem Desktop finden Sie zwei weitere nützliche Icons: Mit *OpenVAS GTK Client* öffnen Sie den in Kapitel 3 beschriebenen Client, mit *Live Installer* installieren Sie die Live-CD auf dem ausführenden System.

Auf der Website der OpenVAS-Entwickler finden Sie übrigens weitere Live-System-Varianten (*http://www.openvas.org/vm.html*), die speziell für die Ausführung mit VirtualBox, VMware und Xen konzipiert sind.

Index

G

H

I

J

K

L

M

N

O

P

R

S

T

V

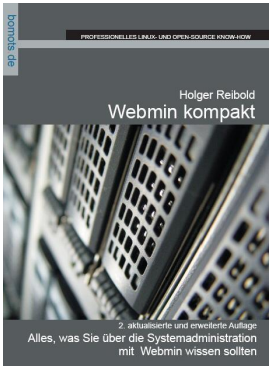
W

X

Y

Z

Weitere Brain-Media.de-Bücher



Linux User und Linux intern empfehlen:
Webmin kompakt
Webmin ist der Klassiker unter den Administrationswerkzeugen für Linux-Systeme. In der stark erweiterten Neuauflage unseres Klassikers zeigen wir Ihnen, wie Sie mit Webmin 1.3.x arbeiten.

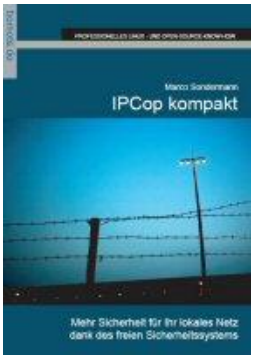
Umfang: 423 Seiten
Preis: 24,80 EUR
ISBN: 978-3-939316-10-7

T3N und Linux Magazin empfehlen:

Guerilla-Marketing mit Open-Source-Tools

Unternehmen mit kleinen Budgets müssen Wege suchen, wie sie durch geschicktes Agieren Neukunden, Multiplikatoren etc. für sich gewinnen. In diesem Buch werden die wichtigsten Ansätze und deren praktische Umsetzung beschrieben.

Umfang: 260 Seiten
Preis: 19,80 EUR
ISBN: 978-3-939316-29-9



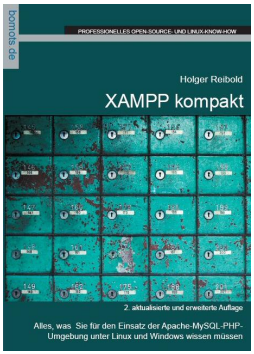
Buchtipp aus Linux-User, entwickler-Magazin und T3N-Magazin:

IPCop kompakt

Das Administrator- und Anwenderhandbuch

IPCop-Guru Marco Sondermann lässt Sie an seiner jahrelangen Erfahrung teilhaben und verrät Ihnen, wie Sie IPCop professionell nutzen.

Umfang: 340 Seiten
Preis: 24,80 EUR
ISBN: 978-3-939316-41-1



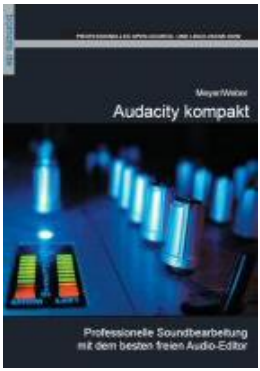
iX empfiehlt:

XAMPP kompakt

(2., akt. u. erw. Auflage)

Mithilfe von XAMPP lässt sich mit minimalem Aufwand eine Apache-MySQL-Perl-PHP-Umgebung aufsetzen. Unser aktualisiertes und erweitertes Anwenderhandbuch zeigt, was Sie dafür wissen sollten.

Umfang: 250 Seiten
Preis: 19,80 EUR
ISBN: 978-3-939316-35-0

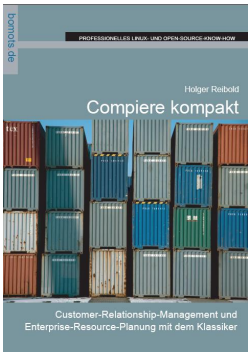


Handling und Funkempfang empfehlen:

Audacity kompakt

Audacity ist ein professioneller Audioeditor, mit dem Sie beispielsweise beliebig viele Spuren mischen und bearbeiten oder Ihre Schallplatten digitalisieren können. Das Buch ist mit Unterstützung eines Core-Entwicklers entstanden.

Umfang: 140 Seiten
Preis: 16,80 EUR
ISBN: 978-3-939316-23-7

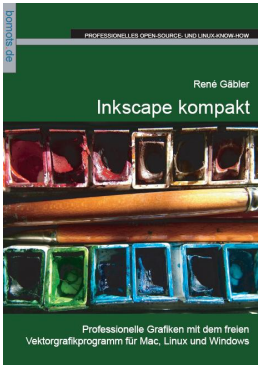


iX empfiehlt:

Compiere kompakt

Compiere gilt als die beste CRM- und ERP-Software der Open-Source-Gemeinde. Sie deckt alle Anforderungen von kleinen und mittleren Betrieben ab. Doch in der Praxis erweist sich Compiere als schwer zu handhaben. Unser Handbuch ebnet Ihnen den erfolgreichen Einstieg.

Umfang: 260 Seiten
Preis: 19,80 EUR
ISBN: 978-3-939316-11-4

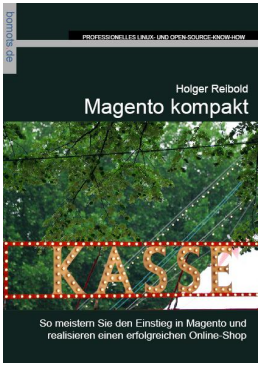


ct- und Mac Life-Buchtipp:

Inkscape kompakt (Neuauflage)

Inkscape ist ein sensationelles freies Vektorgrafikprogramm für Mac, Linux und Windows. In der Neuauflage von „Inkscape kompakt" führt der Autor Sie praxisbezogen in alle wichtigen Aktionen ein. Auch erweiterte Themen wie der Import und Export sowie der Umgang mit dem XML-Editor haben ihren Platz.

Umfang: 180 Seiten
Preis: 16,80 EUR
ISBN: 978-3-939316-45-9

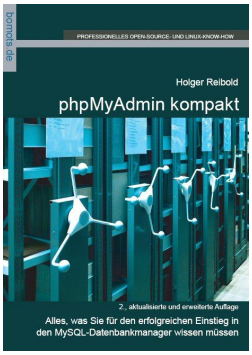


IT-Mittelstand empfiehlt:

Magento kompakt

Magento ist der neue Stern unter den E-Commerce-Lösungen. Endlich gibt es eine komfortable Shop-Lösung, die eine hohe Skalierbarkeit aufweist und sogar Marketingfunktionen integriert. Das notwendige Know-how für den erfolgreichen Einstieg liefert unser Handbuch "Magento kompakt".

Umfang: 280 Seiten
Preis: 19,80 EUR
ISBN: 978-3-939316-56-5



Pro-Linux-, ix- und Linux-Enterprise-Buchtipp:

phpMyAdmin kompakt

(2., akt. und erweiterte Auflage)

phpMyAdmin ist das wichtigste Hilfsmittel für die Administration von MySQL-Servern. Unser Buch zeigt, wie Sie typische Aufgaben im Datenbankalltag bewältigen, Daten und Strukturen erzeugen, Inhalte importieren und exportieren, MySQL-Server verwalten und vieles mehr.

Umfang: 200 Seiten
Preis: 19,80 EUR
ISBN: 978-3-939316-43-5

Reelbox kompakt

Die Reelbox ist die Nummer eins unter den Linux-basierten Festplatten-Receivern. Aber das All-in-one-Mediacenter hat noch mehr zu bieten. Wie Sie das System ausreizen, erfahren Sie in unserem Handbuch, das in Zusammenarbeit mit dem Hersteller entstanden ist.

Umfang: ca. 250 Seiten
Preis: ca. 24,80 EUR
ISBN: 978-3-939316-63-3
erscheint ca. 3. Quartal 2010