OpenVAS Kompendium

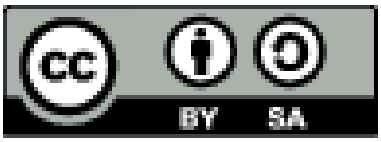
Eine Veröffentlichung des OpenVAS Projektes

Autoren der englischen Version: Jan-Oliver Wagner, Michael Wiegand, Tim Brown, Carsten Koch Mauthe

Übersetzung ins Deutsche: Michael Wiegand, Felix Wolfsteller

Version 1.0.1 vom 20090115

Impressum



Vorwort zur deutschen Version

Dieses Kompendium basiert auf dem OpenVAS-Kompendium Version 1.0.1 in englischer Sprache und wurde zuletzt am 15.01.2009 aktualisiert. Es wurde mit Ausnahme des Abschnittes 11 von Michael Wiegand übersetzt. Als Autoren der einzelnen Abschnitte sind weiterhin die Autoren der englischen Original-Version angegeben. Änderungen werden vom OpenVAS-Team im Allgemeinen zuerst in der englischen Fassung eingebracht. Auch wenn versucht wird, die deutsche und englische Fassung stets auf dem gleichen Stand zu halten, kann es so passieren, dass an der englischen Fassung bereits fortgeschrieben wurde und in ihr neuere Informationen enthalten sind.

Die Übersetzer

Michael Wiegand

und

Felix Wolfsteller

Inhaltsverzeichnis

1 Einführung

1.1 Über dieses Kompendium

(von Jan-Oliver Wagner)

Dieses Kompendium wurde von Teilnehmern des OpenVAS Projektes zusammengestellt. Es soll auf verständliche Weise alle Aspekte des Network Security Scanning mit OpenVAS dokumentieren.

Die Themen dieses Kompendiums reichen von Anleitungen zur Benutzung des OpenVAS-Clients über die Auswahl spezieller Tests und die Erstellung eigener Sicherheitstests in NASL bis zu Details des internen Aufbaus des Scanservers.

Das OpenVAS-Kompendium wird ständig verbessert und erweitert. Es kann vorkommen, dass einige Teile Ihnen nicht ausführlich genug erscheinen und andere Teile komplett fehlen.

Zusätzliche Autoren sind herzlich eingeladen bei der Verbesserung und Erweiterung des Kompendiums zu helfen. Falls Ihnen ein Bereich auffällt, zu dem Sie Ihr Wissen und Ihre Erfahrungen beitragen möchten, koordinieren Sie Ihre Ergänzung bitte mit dem OpenVAS Team. Es ist sinnvoll, diese Koordinierung durchzuführen, *bevor* Sie mit dem Schreiben beginnen; eventuell bearbeitet schon jemand den Bereich, den Sie sich ausgesucht haben.

Die Formatierungssprache für dieses Dokument ist LaTeXmit HyperLaTeX-Erweiterungungen. Die Quellen für dieses Dokument sind in der OpenVAS Quellcodeverwaltung [1] als Modul „openvas-compendium“ verfügbar.

1.2 Über das OpenVAS-Projekt

(von Michael Wiegand)

Der Name „OpenVAS“ steht für „Open Vulnerability Assessment System“ Dieses System stellt eine umfassende Sammlung von Werkzeugen für die Sicherheitsanalyse in Netzwerken zur Verfügung, bietet eine grafische Benutzungsoberfläche und bindet eine Vielzahl von weiteren Sicherheitsanwendungen ein. Das Herzstück des Systems bildet eine Serverkomponente, die eine Sammlung von NVTs (Network Vulnerability Tests) nutzt, um Sicherheitsprobleme in Netzwerksystem und -anwendungen aufzuspüren.

Das Team der OpenVAS-Entwickler besteht sowohl aus verschiedenen Gruppen in Wirtschaft und Forschung als auch aus einzelnen Entwicklern. Die Mehrheit des Entwicklerteams verfügt über langjährige Berufserfahrung im Bereich der IT-Sicherheit und/oder der Softwareentwicklung.

Das OpenVAS-Projekt ist offen für neue Mitglieder. Formalisierte Prozesse an bestimmten Stellen sollen dabei helfen, sich durch die Teilnahme an diesen Prozessen (wie etwa Design, Entwicklung, Test) schnell in das Projekt hineinzufinden.

Alle Bestandteile von OpenVAS sind Freie Software und unter der GNU General Public License (GPL) lizenziert.

OpenVAS hat seine Wurzeln im leider proprietär gewordenen Nessus-Projekt; seit der Abspaltung (fork) von Nessus wird OpenVAS eigenständig weiterentwickelt.

[1] http://wald.intevation.org/projects/openvas

1.3 Über die OpenVAS Software

(von Michael Wiegand)

Die OpenVAS Software besteht aus fünf Modulen, die vom OpenVAS-Projekt zur Verfügung gestellt und gewartet werden. Diese fünf Bestandteile sind:

OpenVAS-Server: Dieses Modul ist der Kern der OpenVAS Software. Es bietet die Möglichkeit, eine große Anzahl von Rechnern in kurzer Zeit zu testen. Scans werden immer von dem Rechner ausgehen, auf dem OpenVAS-Server läuft; aus diesem Grund muss dieser Rechner in der Lage sein, den oder die Zielrechner zu erreichen.

Der Server benötigt drei weitere Module:

OpenVAS-Libraries: Dieses Modul enthält die von OpenVAS genutzten Bibliotheken.

OpenVAS-LibNASL: Die einzelnen Sicherheitstests (Network Vulnerability Tests oder NVTs) sind in der „Nessus Attack Scripting Language“ (NASL) geschrieben. Durch dieses Modul kann OpenVAS-Server diese Skripte lesen und ausführen.

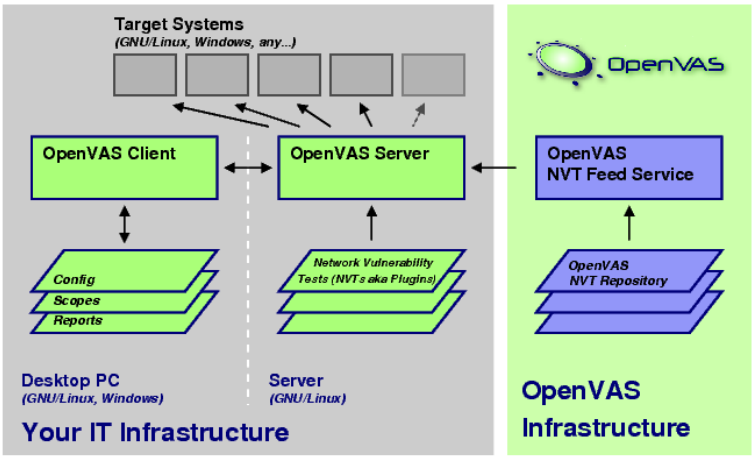
OpenVAS-Plugins: Dieses Modul bietet eine Grundausstattung an NVTs. Bitte beachten Sie, dass der Updatezyklus dieses Moduls nicht dazu gedacht ist, die Verfügbarkeit aktueller NVTs sicherzustellen. Falls Sie aktuelle NVTs benötigen, sollten Sie einen NVT Feed abonnieren, wie in Abschnitt 2.1 beschrieben.

OpenVAS-Client: OpenVAS-Client steuert den OpenVAS Server, verarbeitet die Scanergebnisse und zeigt sie dem Benutzer an. OpenVAS-Client kann auf jedem Rechner laufen, der eine Verbindung zum OpenVAS Server aufbauen kann und kann eine Vielzahl von Servern steuern.

2 Planung eines OpenVAS-basierten Netzwerk Audits

(von Jan-Oliver Wagner)

2.1 Analyse der Abdeckung der verfügbaren Sicherheitstests



Als einen der ersten Schritte bei der Planung eines „Security Audits“ (Sicherheitsprüfung) mit OpenVAS sollten Sie sicherstellen, dass die potentiellen Sicherheitslücken auf den von Ihnen ausgewählten Zielrechnern von den derzeit verfügbaren Sicherheitstests für OpenVAS abgedeckt werden.

Berücksichtigen Sie bitte, dass Releases von OpenVAS in der Regel nur eine Grundausstattung von Sicherheitstests im Form des Moduls `openvas-plugins` mitbringen. Der Updatezyklus dieser Grundausstattung ist eher langsam im Vergleich zum Auftreten neuer Sicherheitslücken und dem Erscheinen neuer NVTs. Über so genannte „Feed Services“ sind neue und aktualisierte NVTs dagegen innerhalb kurzer Zeit erhältlich.

Wenn Sie Ihre Netzwerk auf aktuelle Sicherheitslücken überprüfen wollen, hängt ein erfolgreiches Ergebnis in hohem Maße von der Qualität der Feed Services ab, die Sie verwenden. Obwohl das Modul `openvas-plugins` eine Auswahl von NVTs mitbringt, die eine große Bandbreite älterer, bekannter Schwachstellen abdeckt, wird es zum Zeitpunkt Ihres Scans sehr wahrscheinlich bereits veraltet sein und keine Tests für Sicherheitslücken enthalten, die erst vor kurzer Zeit entdeckt wurden. Wenn Sie auf dem aktuellen Stand bleiben wollen, benötigen Sie einen Feed Service, der Sie mit aktuellen Tests für gerade entdeckte Sicherheitslücken versorgt.

Das OpenVAS-Projekt stellt einen eigenen Feed Service zur Verfügung:

http://www.openvas.org/openvas-nvt-feed.html

Um festzustellen, ob Sie einen (und wenn ja, welchen) aktuellen Feed Service benötigen, sollten Sie die folgenden Fragen bedenken?

- Wie gut decken die über den Feed Service verfügbaren Tests und die von ihm bereitgestellten Familien die von Ihnen ausgewählten Zielrechner ab?
- Falls Sie eine permanente Lösung für Sicherheitsanalysen und nicht nur eine einzelne Überprüfung planen: Wie zuverlässig ist der Feed Service organisiert?
- Sind die Signaturen der NVTs vertrauenswürdig genug, um die von Ihnen benötigte Qualität sicherzustellen?

2.2 Auswahl des Standortes für einen Scanserver

Wenn Sie den Einsatz von OpenVAS in Ihrem Netzwerk planen, hängt der ideale Standort Ihres Servers davon ab, welche Art von Zielrechnern Sie überprüfen möchten.

- Der Zielrechner ist öffentlich zugänglich:

  Eine Anzahl von NVTs sind darauf ausgelegt, den gleichen Ursprung wie verschiedene echte Angriffe zu haben: Ein entferntes Netzwerk. Falls Sie hauptsächlich an diesen Tests interessiert sind, können Sie einen beliebigen Standort außerhalb des Zielnetzwerks für Ihren OpenVAS Server auswählen.

  Wir raten Ihnen allerdings, vor einem Scan die Systemadministration des Zielsystems zu kontaktieren und sie darüber zu informieren, dass Sie einen Scan mit OpenVAS planen. Da OpenVAS auf dem Zielsystem aktiv nach Sicherheitslücken sucht, kann ein Scan unter gewissen Umständen wie ein Angriff auf das Zielsystem aussehen und so rechtliche oder technische Schritte von Seiten der Systemadministration des Zielsystems nach sich ziehen.
- Der Zielrechner befindet sich in einem abgeschlossenen Intranet:

  In diesem Fall sollten Sie Ihren Scan mit Ihrer Systemadministration koordinieren.

  Abhängig vom Aufbau Ihres Intranets müssen Sie eventuell evaluieren, wie Sie alle Zielrechner von Ihrem OpenVAS Server aus erreichen können. In manchen Fällen kann die Installation von mehreren Servern sinnvoll sein, um eine vollständige Abdeckung Ihres Netzwerks sicherzustellen.

  Falls Sie lokale Sicherheitstests (Local Security Checks, LSCs) nutzen möchten, müssen Sie die Zielrechner für einen Fernzugriff vorbereiten. Für Unix-basierte Systeme ist für die Ausführung von LSCs im Allgemeinen ein Zugriff per ssh nötig, bei Microsoft Windows-basierten Systemen geschieht dies über SMB Freigaben.

2.3 Auswahl der Ausstattung eines Scanservers

2.3.1 Hardware

OpenVAS stellt unter normalen Umständen keine herausragenden Anforderungen an die Leistungfähigkeit des Scanservers, ein handelsüblicher Server sollte in den meisten Fällen ausreichen.

Die meisten Benutzer nutzen OpenVAS in einer IA32-Architektur. 64bit- und PowerPC-basierte Architekturen werden ebenfalls von OpenVAS unterstützt, werden aber nicht im gleichen Ausmaß getestet.

2.3.2 Betriebssystem

Für OpenVAS-Server werden ausschließlich Unix-basierte Betriebssysteme unterstützt. Von diesen Systemen sind Linux-basierte System am umfassendsten getestet. BSD-basierte Systeme werden ebenfalls unterstützt, aber nicht im gleichen Umfang getestet.

Bei der Auswahl des Betriebssystems sollten Sie folgende Aspekte berücksichtigen:

- OpenVAS kann entweder aus dem Quelltext kompiliert werden oder aus vorbereiteten Binärpaketen installiert werden. Falls Sie die letztere Methode vorziehen, sollten Sie sich über die Verfügbarkeit von Binärpaketen für die verschiedenen Betriebssysteme und Distributionen informieren.
- Verschiedene Sicherheitswerkzeuge können in OpenVAS integriert werden. Falls Sie diese Werkzeuge nutzen möchten, sollten Sie sicherstellen, dass die Werkzeuge für das von Ihnen gewählte Betriebssystem verfügbar sind. Auch hier sind Binärpakete auf manchen Systemen besser verfügbar als auf anderen.

3 OpenVAS-Server installieren und konfigurieren

(von Tim Brown, Jan-Oliver Wagner und Michael Wiegand)

3.1 Binärpakete installieren

Binärpakete für die verbreitetsten Linux-Distributionen und einige andere Unix-basierte Systeme sind über die OpenVAS-Website verfügbar.

Bitte beachten Sie, dass der Umfang der automatischen Vorkonfiguration bei der Installation des Paketes von der verwendeten Distribution und den bei dieser Distributionen für die Paketierung Verantwortlichen abhängt. Eine komplette Konfigurationsanleitung finden Sie im Abschnitt 3.3.

Falls keine (oder nicht alle) OpenVAS-Module für das Betriebssystem oder die Distribution Ihrer Wahl verfügbar sind, ist es auf den meisten Unix-basierten Systemen in der Regel möglich, OpenVAS selbst zu kompilieren (siehe Abschnitt 3.2). Bei Bedarf können Sie gerne die bei dieser Distribution für die Paketierung Verantwortlichen darauf hinweisen, dass Sie an OpenVAS-Paketen interessiert sind und dass das OpenVAS Entwickler-Team neue Paketierungen nach Möglichkeit unterstützt.

3.1.1 Debian und Ubuntu

OpenVAS-Server wird zur Zeit in Debian und Ubuntu integriert. Das bedeutet, dass Sie OpenVAS-Server in Zukunft einfach mittels des `apt-get` Mechanismus installieren können.

Während der Integration in Debian und Ubuntu kann es vorkommen, dass noch nicht alle Teile von OpenVAS in den für diese Distributionen offiziellen Paketquellen verfügbar sind; eventuell sind die fehlenden Teile aber über andere Paketquellen verfügbar. Auf der OpenVAS-Website finden Sie aktuelle Informationen über die verfügbaren Pakete und Anleitungen zur Installation unter Debian und Ubuntu.

3.1.2 Gentoo

OpenVAS Pakete für Gentoo sind über das „Portage“-System verfügbar. In der Gentoo-Dokumentation und auf der OpenVAS-Website finden Sie nähere Informationen zur Installation dieser Pakete.

3.1.3 RPM-basierte Distributionen

Pakete für einige Versionen von OpenSUSE, Fedora, Mandriva und anderen RPM-basierten Distributionen sind in inoffiziellen Paketquellen unter

http://download.opensuse.org/repositories/security:/openvas:/ und http://www.atomicorp.com/channels/atomic/

verfügbar. In der Dokumentation Ihrer Distributionen finden Sie eine Anleitung zum Hinzufügen von Paketquellen und zum Installieren dieser Pakete.

3.1.4 FreeBSD

Die FreeBSD Ports- und Paketesammlung enthält Ports und Pakete für alle OpenVAS-Module.

Mit den folgenden Befehlen können Sie die FreeBSD Ports auf Ihrem FreeBSD System kompilieren und installieren:

```
cd /usr/ports/security/openvas-libraries/ && make install clean
cd /usr/ports/security/openvas-libnasl/ && make install clean
cd /usr/ports/security/openvas-server/ && make install clean
cd /usr/ports/security/openvas-plugins/ && make install clean
```

Falls Sie die Binärpakete vorziehen, so können Sie diese mit den folgenden Befehlen installieren:

```
pkg_add -r openvas-libraries
pkg_add -r openvas-libnasl
pkg_add -r openvas-server
pkg_add -r openvas-plugins
```

3.2 OpenVAS-Server kompilieren

3.2.1 Aktuelles Quelltext-Release

Auf der OpenVAS-Website finden Sie den Link zum aktuellen Quelltext-Release.

Wenn Sie ein Quelltext-Release nutzen möchten, sollten Sie bedenken, dass dabei eventuell im Laufe der Installation Dateien an andere Orte kopiert werden als die von Ihnen oder Ihrer Distribution erwarteten Orte im Dateisystem. Dadurch kann es durch Unachtsamkeit leicht zu unerwarteten Ergebnissen kommen, falls Sie versuchen, Releases einer anderen Serie zu installieren oder alte Versionen von OpenVAS zu entfernen. Aus diesem Grund empfehlen wir Ihnen, die Module mit einem Präfix zu konfigurieren (bspw. `./configure prefix=/opt/openvas`) und für verschieden Serien unterschiedlich Präfixe zu benutzen. Auf diese Weise können Sie die zu einer bestimmten OpenVAS-Version gehörenden Dateien einfach isolieren; falls Sie eine bestimmte OpenVAS-Installation entfernen möchten, können Sie einfach das jeweilige Unterverzeichnis löschen.

Laden Sie zunächst die vier Quelltextarchive mit der Endung „.tar.gz“ herunter und entpacken Sie sie mit dem Befehl `tar -xzf openvas-MODUL-N.N.N.tar.gz`. Das Kompilieren des Quelltextes ist zur Zeit auf GNU/Linux-Systeme ausgerichtet, funktioniert aber eventuell in anderen Umgebungen.

Sie müssen die Module in der folgenden Reihenfolge kompilieren und installieren:

1. openvas-libraries
2. openvas-libnasl
3. openvas-server
4. openvas-plugins

In der Datei `INSTALL_README` im Verzeichnis „openvas-libraries“ finden Sie eine Anleitung für die nächsten Schritte.

Wiederholen Sie dies nun für die übrigen Module und lesen Sie die jeweiligen `INSTALL`- und/oder `README`-Dateien.

3.2.2 Aktueller Entwicklungsstand (direkt aus der Quelltextverwaltung)

Sie benötigen das Programm „subversion“, um den aktuellen Entwicklungsstand herunterzuladen:

```
$ svn checkout
https://svn.wald.intevation.org/svn/openvas/trunk/openvas-libraries
$ svn checkout https://svn.wald.intevation.org/svn/openvas/trunk/openvas-libnasl
$ svn checkout https://svn.wald.intevation.org/svn/openvas/trunk/openvas-server
$ svn checkout https://svn.wald.intevation.org/svn/openvas/trunk/openvas-plugins
```

In der Datei `INSTALL_README` im Verzeichnis „openvas-libraries“ finden Sie eine Anleitung für die nächsten Schritte.

Wiederholen Sie dies nun für die übrigen Module und lesen Sie die jeweiligen `INSTALL`- und/oder `README`-Dateien.

Das Team der OpenVAS-Entwickler fühlt sich dazu verpflichtet, eine möglichst hohe Codequalität zu gewährleisten. Bitte beachten Sie trotzdem, dass Sie einen aktuellen Entwicklungsstand benutzen, der unvollständig und instabil sein könnte und nicht in Produktivumgebungen eingesetzt werden sollte.

3.3 OpenVAS-Server konfigurieren

(von Michael Wiegand)

Nachdem Sie OpenVAS-Server installiert haben, sind einige zusätzliche Schritte notwendig, damit Sie Ihre OpenVAS-Installation effektiv nutzen können: So müssen Sie beispielsweise Zertifikate für den Server erstellen und Benutzer anlegen, bevor Sie sich mit dem Server verbinden können. Beachten Sie, dass die folgenden Anweisungen sich sowohl auf die Serie 1.0 als auch auf die Serie 2.0 beziehen.

Falls Sie OpenVAS aus Pakete installiert haben, die von Ihrer Distribution zur Verfügung gestellt wurden, müssen Sie unter Umständen andere Schritte als die in diesem Kompendium beschriebenen durchführen. Bitte konsultieren Sie die Dokumentation zur Ihrer Distribution, falls Sie nähere Informationen benötigen.

Bitte beachten Sie außerdem, dass Sie eventuell zusätzliche Software installieren müssen, falls Sie in Kapitel 8 beschrieben Sicherheitswerkzeuge zusammen mit OpenVAS nutzen wollen.

3.3.1 Erstellung eines Serverzertifikates

Aus Sicherheitsgründen ist eine Kommunikation zwischen dem OpenVAS Server und dem Client nur über eine SSL-verschlüsselte Verbindung möglich. Damit eine verschlüsselte Verbindung aufgebaut werden kann, muss der Server ein gültiges SSL-Zertifikat besitzen. Falls für den Rechner, auf dem der OpenVAS Server läuft, noch kein Zertifikat verfügbar ist, müssen Sie ein neues erstellen.

Am einfachsten geht dies mit dem im OpenVAS-Server Modul enthaltenen Skript `openvas-mkcert`. Dieses Skript erstellt zwei Zertifikate: Ein Zertifikat für eine lokale Zertifizierungsstelle (Certificate Authority, CA) und ein zweites Zertifikat für den OpenVAS Server, das von dieser CA signiert ist und beim Verbindungsaufbau übermittelt wird.

Falls Sie bereits über eine X.509-basierte Public-Key-Infrastruktur (PKI) verfügen, können Sie selbstverständlich auch ein Zertifikat verwenden, das von der zuständigen CA (etwa Ihrem Trust-Center) signiert wurde.

3.3.2 Neue Benutzer hinzufügen

Um einen OpenVAS Server nutzen zu können, muss der Benutzer auf dem Server ein Benutzerkonto besitzen. Das OpenVAS-Server Modul erhält hierfür das Skript `openvas-adduser`. Bei der Erstellung eines Benutzerkontos mit `openvas-adduser` können Sie angeben, ob sich der Benutzer mit einem Passwort oder mit einem Zertifikat authentifizieren soll und optional die Zugriffsrechte dieses Benutzers einschränken.

Beschränkte Zugriffsrechte können nützlich sein, um zu verhindern, dass Benutzer beliebige Zielrechner scannen können. Sie können Regeln angeben, die Benutzer auf bestimmte Zielrechner oder -netzwerke beschränken oder einen Benutzer nur das Scannen des eigenen Rechners erlauben.

Der korrekte Syntax für die benutzerspezifischen Zugriffsregeln ist wie folgt:

```
accept|deny ip/mask
```

und

```
default accept|deny
```

Wobei `mask` die Netzmaske des Zugriffsregel nach dem CIDR-Schema ist.

Die `default` Regel muss die letzte Regel sein und definiert die grundlegende Richtlinie für diesen Benutzer.

Die folgenden Regeln beispielsweise erlauben es dem Benutzer, ausschließlich die Netzwerke 192.168.1.0/24, 192.168.3.0/24 und 172.22.0.0/16 zu scannen:

```
accept 192.168.1.0/24
accept 192.168.3.0/24
accept 172.22.0.0/16
default deny
```

Die folgenden Regeln erlauben es dem Benutzer, beliebige Rechner und Netzwerke zu scannen, bis auf das Subnetz 192.168.1.0/24:

```
deny 192.168.1.0/24
default accept
```

Das Schlüsselwort `client_ip` wird zur Laufzeit durch die IP-Adresse des Benutzers ersetzt. Falls den Benutzer auf das System beschränken wollen, von dem aus er sich mit dem Server verbindet, können Sie die folgenden Regeln verwenden:

```
accept client_ip
default deny
```

3.3.3 Fortgeschrittene Konfiguration

Falls Sie die voreingestellte Konfiguration von OpenVAS-Server ändern möchten, können Sie dies in der Konfigurationsdatei tun, die Sie im Normalfall unter `/etc/openvas/openvasd.conf` finden.

Die folgenden Einstellungen können in der `openvasd.conf` vorgenommen werden. Bitte beachten Sie, dass die Voreinstellungen für Ihre Distribution von den hier beschriebenen Werten abweichen können.

plugins_folder Der Pfad, indem die NVT Skripte gefunden werden können.

(Voreinstellung: /lib/openvas/plugins)

max_hosts Die maximale Anzahl von Zielrechnern, die auf einmal getestet werden können.

(Voreinstellung: 30)

max_checks Die maximale Anzahl von Tests, die gleichzeitig gegen einen Zielrechner durchgeführt werden.

(Voreinstellung: 10)

be_nice Priorität der Serverprozesses („Niceness“). Wenn diese Einstellung auf dem Wert „yes“ steht, wird `openvasd` seine Prozesspriorität auf 10 herabsetzen.

(Voreinstellung: no)

logfile Die Datei, in die der OpenVAS-Server Meldungen schreiben soll. Wenn dieser Wert auf `syslogd` gesetzt wird, wird OpenVAS-Server den `syslogd`-Dienst benutzen.

(Voreinstellung: /var/log/openvas/openvasd.messages)

log_whole_attack Diese Einstellung bestimmt, wie detailliert der Scanablauf protokolliert werden soll. Wenn diese Einstellung auf „no“ gesetzt ist, wird lediglich der Start- und Endzeitpunkt des Scans protokolliert. Falls diese Einstellung auf „yes“ gesetzt wird, protokolliert OpenVAS-Server den Ablauf detaillierter und protokolliert beispielsweise die Zeit, die die einzelnen NVTs zur Ausführung benötigt haben. Bitte beachten Sie, dass OpenVAS-Server dadurch mehr Festplattenkapazität in Anspruch nehmen wird und während des Scanvorgangs öfter auf die Festplatte zugreifen wird.

(Voreinstellung: no)

log_plugins_name_at_load Diese Einstellung bestimmt, ob die Namen der NVTs, die vom Server geladen werden, protokolliert werde sollen.

(Voreinstellung: no)

dumpfile Diese Einstellung konfiguriert den Name der Datei, die für Debug-Ausgaben genutzt werden soll. Wenn diese Einstellung auf „-“ gesetzt wird, werden Debug-Ausgaben in Standardausgabe (stdout) geschrieben.

(Voreinstellung: /var/log/openvas/openvasd.dump)

rules Der Dateiname für die Zugriffsregeln.

(Voreinstellung: /etc/openvas/openvasd.rules)

users Der Dateiname für die Benutzerdatenbank.

(Voreinstellung: /etc/openvas/openvasd.users)

cgi_path Der voreingestellte Pfad für zu überprüfende CGI-Skripte; mehrere Werte werden durch Doppelpunkte („:“) getrennt.

(Voreinstellung: /cgi-bin:/scripts)

port_range Der Bereich der Ports, die von den Portscannern gescannt werden sollen. Wenn diese Einstellung auf „default“ gesetzt ist, wird OpenVAS-Server die in der Datei `/var/lib/openvas/openvas-services` aufgelisteten Ports überprüfen.

(Voreinstellung: default)

optimize_test Manche Sicherheitstests geben an, dass sie nur gestartet werden sollen, wenn bestimmte Werte bereits in der Wissensbasis vorhanden sind oder wenn ein bestimmter Port offen ist. Wenn diese Option aus „yes“ gesetzt wird, beschleunigt dies den Scan, kann aber dazu führen, dass einige Sicherheitslücken nicht gefunden werden.

(Voreinstellung: yes)

language Die Sprache, die in den NVT-Beschreibungen verwendet werden soll. Zur Zeit werden die Werte „english“ und „french“ unterstützt.

(Voreinstellung: english)

checks_read_timeout Die Zeitbeschränkung (in Sekunden) für Lesevorgänge auf Sockets während des Scanvorgangs.

(Voreinstellung: 5)

non_simult_ports Mit dieser Option kann eine Liste von Ports angegeben werden, gegen die NVTs nicht gleichzeitig laufen sollen.

(Voreinstellung: 139, 445)

plugins_timeout Die maximale Ausführungszeit für ein NVT (in Sekunden).

(Voreinstellung: 320)

safe_checks Manche Sicherheitstests können den Zielrechner schädigen, etwa durch einen Störung eines bestimmten Dienst bis zu einem Neustart des Dienstes oder des Zielrechners. Wenn diese Einstellung auf „yes“ steht, wird sich OpenVAS-Server auf die Banner der entsprechenden Dienste verlassen anstatt den Test wirklich durchzuführen. Dies führt zu einem weniger zuverlässigen Bericht, macht aber eine Einschränkung der Funktionalität des Zielrechners während des Scanvorgangs weniger wahrscheinlich.

(Voreinstellung: yes)

auto_enable_dependencies Wenn dieses Einstellung auf „yes“ gesetzt ist, wird OpenVAS-Server automatisch NVTs aktivieren, von denen die vom Benutzer ausgewählten NVTs abhängig sind.

(Voreinstellung: yes)

silent_dependencies Wenn diese Einstellung auf „yes“ gesetzt ist, wird die Ausgabe von automatisch aktivierten NVTs nicht an den Client gesendet.

(Voreinstellung: yes)

use_mac_addr Verwendet MAC-Adressen und nicht IP-Adressen, um Rechner zu identifizieren; dies kann etwa in DHCP-Netzwerken nützlich sein.

(Voreinstellung: no)

save_knowledge_base Diese Einstellung steuert, ob die während des Scanvorgangs erstellte Wissensbasis nach dem Scan auf dem Server gespeichert werden soll.

(Voreinstellung: no)

kb_restore Diese Einstellung steuert, ob die Wissensbasis für einen erneuten Test wieder geladen werden soll.

(Voreinstellung: no)

only_test_hosts_whose_kb_we_dont_have Wenn diese Einstellung aktiviert ist, scannt OpenVAS nur die Rechner, die noch nicht in der Wissensbasis vorhanden sind. Diese Einstellung kann beispielsweise eingesetzt werden, um Rechner einmal zu scannen, wenn Sie das erste Mal im Subnetz erscheinen.

(Voreinstellung: no)

only_test_hosts_whose_kb_we_have Wenn diese Einstellung aktiviert ist, wird OpenVAS nur die Rechner scannen, die bereits in der Wissensbasis vorhanden sind. Dies kann nützlich sein, wenn eine bestimmte Menge von Zielrechnern regelmäßig überprüft werden soll.

(Voreinstellung: no)

kb_dont_replay_scanners Wenn sowohl diese Einstellung als auch die Einstellung `kb_restore` aktiviert ist, werden Scanner-NVTs nicht erneut gestartet falls sie bereits in der Vergangenheit gestartet wurden.

(Voreinstellung: no)

kb_dont_replay_info_gathering Wenn sowohl diese Einstellung als auch die Einstellung `kb_restore` aktiviert ist, werden NVTs zur Informationssammlung nicht erneut gestartet falls sie bereits in der Vergangenheit gestartet wurden.

(Voreinstellung: no)

kb_dont_replay_attacks Wenn sowohl diese Einstellung als auch die Einstellung `kb_restore` aktiviert ist, werden Angriffs-NVTs nicht erneut gestartet falls sie bereits in der Vergangenheit gestartet wurden.

(Voreinstellung: no)

kb_dont_replay_denials Wenn sowohl diese Einstellung als auch die Einstellung `kb_restore` aktiviert ist, werden „Denial of Service“-NVTs nicht erneut gestartet falls sie bereits in der Vergangenheit gestartet wurden.

(Voreinstellung: no)

kb_max_age Diese Option steuert das maximale Alter (in Sekunden) der Wissensbasis.

(Voreinstellung: 864000)

slice_network_addresses Wenn diese Einstellung auf „yes“ gesetzt ist, wird OpenVAS das Netzwerk nicht der Reihe nach scannen (10.0.0.1, 10.0.0.2, 10.0.0.3), sondern wird versuchen, den Scan gleichmäßig über das Netzwerk zu verteilen (etwa: 10.0.0.1, 10.0.0.127, 10.0.0.2, 10.0.0.128).

(Voreinstellung: no)

nasl_no_signature_check Wenn dieses Einstellung auf „yes“ gesetzt ist, wird OpenVAS-Server die kryptografischen Signaturen der NVTs nicht überprüfen und wird NVTs ausführen, auch wenn diese über keine oder keine gültige Signatur verfügen. Berücksichtigen Sie bitte, dass dies eventuell ein Sicherheitsrisiko darstellen kann. Da das Modul openvas-plugins im derzeitigen Entwicklungsstand keine Signaturen enthält, werden Sie nur eine sehr begrenzte Anzahl von Plugins benutzen können, wenn Sie diese Option auf „no“ setzen und Ihre Plugin-Sammlung noch nicht mit eine NVT Feed Service synchronisiert haben, der Signaturen enthält. Aus diesem Grund ist die Voreinstellung für diese Option „yes“, bis Signaturen mit allen Plugins ausgeliefert werden.

(Voreinstellung: yes)

3.4 NVT Feeds konfigurieren

(von Jan-Oliver Wagner und Tim Brown)

In diesem Abschnitt wird beschrieben, wie ein NVT „Feed Service“ (etwa: NVT Abonnement) funktioniert und wie Sie einen Feed Service nutzen können, um Ihre NVT-Sammlung auf dem neuesten Stand zu halten.

Ein OpenVAS NVT Feed Service stellt eine Sammlung von NVTs (etwa in Form von „.nasl“- und „.inc“-Dateien) zur Verfügung, die Sie sich in Ihre OpenVAS-Installation herunterladen können.

Bei einer Aktualisierung Ihrer NVT-Sammlung über einen NVT Feed Service werden nur neue und veränderte NVT 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 von Bedeutung, wenn Sie Ihren OpenVAS Server dazu konfiguriert haben, nur signierte NVTs auszuführen.

3.4.1 Voraussetzungen

Abgesehen von dem `openvas-plugins`-Modul, welches das Skript `openvas-nvt-sync` enthält, müssen auf dem System, auf dem der OpenVAS Server läuft, die Programme `rsync` und `md5sum` verfügbar sein. Falls Sie OpenVAS aus einem Binärpaket installiert haben, sollte die Paketverwaltung Ihrer Distribution die Verfügbarkeit dieser Programme bereits automatisch sichergestellt haben.

3.4.2 Synchronisation mit einem OpenVAS NVT Feed

Um Ihre NVT-Sammlung mit einem OpenVAS NVT-Feed zu synchronisieren, müssen Sie folgende Schritte durchführen:

1. Überprüfen Sie die Einstellungen des Synchronisationsskriptes: In der Regel wird das Skript unter `/usr/sbin/openvas-nvt-sync` installiert.

   Sie sollten überprüfen, ob die Variablen `NVT_DIR` und `FEED` die für Ihre Konfiguration korrekten Werte beinhalten. Für `NVT_DIR` sollte dies der Fall sein, sofern Sie nicht von dem beschrieben Installationsablauf abgewichen sind. Für `FEED` existiert momentan nur der voreingestellte Feed Service; hier ist im Allgemeinen keine Änderung notwendig.

2. Rufen Sie das Synchronisationsskript auf:

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

   Das Skript synchronisiert sich nun mit dem angegeben NVT Feed Service. Nach der Synchronisation werden die Prüfsummen aller synchronisierten Datei überprüft. Falls dabei ein Fehler festgestellt wird, erhalten Sie eine Fehlermeldung. In diesem Fall sollten Sie einige Minuten später erneut eine Synchronisation versuchen; meist sind Netzwerkverzögerung oder eine gleichzeitige Aktualisierung der NVT-Sammlung auf dem RSYNC-Server dafür verantwortlich. Sollte der Fehler wiederholt auftreten, melden Sie dies bitte der OpenVAS-Mailingliste. Eine Anleitung zur Automatisierung der Feed-Aktualisierung finden Sie in Kapitel 3.4.4.

3. Starten Sie den OpenVAS Server neu:

   ```
   # kill -1 PID
   ```

   Wobei PID für die Prozess-ID des `openvasd`-Prozesses steht. Sie können ebenfalls den Befehl `killall openvasd` benutzen, falls Sie damit Erfahrung haben.

3.4.3 Verfügbare NVT Feed Services

Das OpenVAS-Projekt stellt unter `rsync://rsync.openvas.org:/nvt-feed` einen NVT Feed Service zur Verfügung. Dieser ist im `openvas-nvt-sync` voreingestellt. Die NVTs sind mit dem Zertifikat für OpenVAS-Transferintegrität signiert.

3.4.4 Automatische Synchronisation mit einem NVT Feed Service

- Erstellen Sie ein Skript mit dem Namen `openvas-update` und speichern Sie es, etwa 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
```

- Fügen Sie zu Ihrer `crontab` die folgende Zeile hinzu:

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

3.5 Verwaltung von NVT Signaturen

(von Jan-Oliver Wagner)

In diesem Abschnitt wird erklärt, was Sie tun müssen, damit ihr OpenVAS-Server ausschließlich signierte NVTs mit einer von Ihnen festgelegten Vertrauensgrad ausführt.

Zur Zeit können Sie signierte NVTs beispielsweise über den im Skript `openvas-nvt-sync` voreingestellten Feed Service erhalten. Die in diesem Feed enthaltenen Signaturen beziehen sich auf das Zertifikat „OpenVAS Transfer Integrity“, das über die OpenVAS-Website unter http://www.openvas.org/trusted-nvts.html verfügbar ist.

3.5.1 Was ist eine Signatur?

Über einen Algorithmus wird eine eindeutige Prüfsumme für eine Datei berechnet. Falls sich auch nur ein Zeichen in der Datei verändert, ändert sich die Prüfsumme ebenfalls. Diese Prüfsumme wird nun auf eine Weise digital signiert, die es Ihnen erlaubt mit einem öffentlichen Zertifikat zu überprüfen, ob diese Signatur mit einem bestimmten Schlüssel durchgeführt wurde. Ein Schlüssel und ein Zertifikat stehen immer in Beziehung zueinander; falls die signierte Datei von einem Dritten modifiziert wurde, wird die Überprüfung der Signatur fehlschlagen. In diesem Fall sollten Sie der Datei nicht vertrauen.

Falls sowohl Datei als auch Signatur intakt sind, bleibt die Frage, ob Sie dem Besitzer dieses Schlüssels vertrauen. Es gibt viele Wege, eine Antwort auf diese Frage zu finden; falls Sie sich dazu entscheiden, dem Besitzer diese Schlüssels vertrauen, können Sie die Datei als vertrauenswürdig akzeptieren.

In anderen Worten: Die Prüfsumme stellt die Integrität der Datei sicher und ändert sich, falls die Datei auf dem Weg zwischen dem NVT Feed Server und Ihrem System verändert wurde. Die Signatur hingegen bescheinigt die Authentizität der Datei – durch die Signatur signalisiert der Verwalter des Feed Service, dass die auf dem Feed Server verfügbare Datei getestet wurde und authentisch ist.

Auf diese Weise können Sie überprüfen, ob die Ihnen vorliegende Datei die gleiche ist, die von dem Verwalter des Feed Service getestet wurde. Es liegt nun in Ihrer Verantwortung zu überprüfen, ob der Verwalter oder die Verwalterin des Feed Services wirklich die Person ist, die er oder sie vorgibt zu sein und ob die von ihm oder ihr durchgeführten Tests für Sie ausreichend sind.

3.5.2 Das Signatur Format

Die Signaturen für OpenVAS-NVTs und mit NVTs verbundenen Dateien („.nasl“, „.inc“) sind so genannte „ASCII-armored detached OpenPGP signatures“, die mit GnuPG erzeugt wurden. Dieses Format bietet die folgenden Vorteile:

- Mehrere Signaturen pro NVT sind möglich
- Administratoren können entscheiden, welchen Schlüsseln sie vertrauen
- Signaturen können mit weitverbreiteten Programmen wie GnuPG erstellt und überprüft werden

- „Detached signatures“ (also Signaturen in separaten Dateien) führen nicht zu Änderungen in den signierten Dateien, wie es integrierte Signaturen tun würden

Der Dateiname der Signatur besteht im Allgemeinen auf dem Name der signierten Datei und dem Zusatz „.asc“. Eine Signatur für die Datei „meinskript.nasl“ wäre also in der Datei „meinskript.nasl.asc“ zu finden.

Bitte beachten Sie an dieser Stelle den Unterschied zu Nessus: Nessus-Signaturen waren integrierte X.509-basierte Signaturen, die nicht die oben genannten Vorteile boten. Beachten Sie, dass OpenVAS keine Nessus-Signaturen unterstützt und NVTs als unsigniert einstufen wird, auch wenn sie über eine gültige Nessus-Signatur verfügen.

3.5.3 Überprüfung von Signaturen

Sie können die Überprüfung der Signaturen durch den OpenVAS Server aktivieren, indem Sie in der Konfigurationsdatei die Einstellung `nasl_no_signature_check` auf „no“ setzen (siehe auch Abschnitt 3.3.3).

Beim Starten des OpenVAS-Dienstes (openvasd) werden alle Signaturen auf ihre Gültigkeit überprüft. Nur vertrauenswürdige Dateien werden vom Server berücksichtigt und an den Client übermittelt.

Für die Überprüfung wird eine vom OpenVAS Server verwaltete Liste von Zertifikaten genutzt. Diese Liste besteht aus einer Datei im „GnuPG Keyring“-Format und liegt in der Regel im Verzeichnis `/etc/openvas/gnupg`.

Wenn OpenVAS-Server die Signatur einer Datei überprüft, werden alle in der entsprechenden Signaturdatei vorhanden Signaturen überprüft; für einen erfolgreiche Überprüfung müssen alle Signaturen vollständig gültig sein. Dies bedeutet, dass alle der folgenden Bedingungen für allen Signaturen, die diese Datei signiert haben, erfüllt werden müssen:

- Das Zertifikat muss im Keyring vorhanden sein.
- Der Schlüssel muss vollständig gültig sein.
- Die Signatur muss gültig sein.

Falls eine der Signaturen nicht alle Bedingungen erfüllt, wird die Datei als nicht vertrauenswürdig eingestuft und nicht ausgeführt. Falls alle Signaturen alle Bedingungen erfüllen, wird die Datei als vertrauenswürdig eingestuft und kann ausgeführt werden. Falls keine Signaturdatei vorhanden ist, wird die Datei nicht ausgeführt.

Bitte beachten Sie auch hier den Unterschied zu Nessus: Bei Nessus-Signaturen wurden drei Vertrauensgrade unterschieden: Keine Signatur, eine ungültige Signatur und ein gültige Signatur. NVTs ohne Signatur wurden in einem abgesicherten („restricted“) Modus ausgeführt und konnten keine als kritisch eingeschätzten Funktionen ausführen. Im Gegensatz dazu unterscheidet OpenVAS nur zwischen vertrauenswürdigen und nicht vertrauenswürdigen Dateien.

3.5.4 Hinzufügen eines Zertifikates

Um ein Zertifikat zu Ihrem OpenVAS-Server Keyring hinzuzufügen, können Sie den folgenden Befehl nutzen:

```
# gpg --homedir=/etc/openvas/gnupg --import certificate-file.asc
```

Verfügbare Zertifikate finden Sie auf der OpenVAS-Website unter http://www.openvas.org/trusted-nvts.html.

3.5.5 Vertrauen festlegen

Um Ihr Vertrauen in einen Schlüssel festzulegen, mit dem NVTs signiert wurden, benötigen Sie einen Schlüssel für Ihre OpenVAS-Installation. Sie können dazu einen existierenden Schlüssel benutzen oder einen neuen Schlüssel erstellen:

```
# gpg --homedir=/etc/openvas/gnupg --gen-key
```

Dieser Schritt muss nur einmal pro OpenVAS-Installation ausgeführt werden.

Damit OpenVAS eine Signatur als vertrauenswürdig einstuft, muss der für die Erstellung der Signatur verwendete Schlüssel gültig sein. Ein sich auf diesen Schlüssel beziehendes Zertifikat, das gerade importiert wurde, hat eine unbekannte Gültigkeit und wird von OpenVAS-Server als ungültig eingestuft.

Um ein Zertifikat als vertrauenswürdig einzustufen müssen Sie es signieren. Es wird empfohlen, für diesen Zweck lokale Signaturen zu verwenden, die lediglich im Keyring Ihre OpenVAS-Installation vorhanden sind.

Um ein Zertifikat signieren zu können, müssen Sie zunächst die `KEY_ID` des Zertifikates wissen. Diese erhalten Sie entweder von der OpenVAS-Website oder mit dem Befehl `list-keys`. Dann können Sie lokal signieren:

```
# gpg --homedir=/etc/openvas/gnupg --list-keys

# gpg --homedir=/etc/openvas/gnupg --lsign-key KEY_ID
```

Vor dem Signieren sollten Sie sich absolut sicher sein, dass Sie das korrekte Zertifikat unterzeichnen. Sie sollten den Fingerabdruck der Zertifikates und andere Methoden nutzen, um sich davon zu überzeugen.

3.5.6 Entfernen eines Zertifikates

```
# gpg --homedir=/etc/openvas/gnupg --delete-keys KEY_ID
```

3.5.7 Manuelle Überprüfung eines Zertifikates

Falls Sie die Gültigkeit einer NVT von Hand überprüfen möchten, können Sie dazu entweder GnuPG verwenden:

```
$ gpg --homedir=/etc/openvas/gnupg gpg --verify script.nasl.asc script.nasl
```

Oder Sie können den eigenständigen NASL Interpreter nutzen:

```
$ openvas-nasl -p script.nasl
```

Die Option „-p“ bedeutet, dass das Skript nur übersetzt, aber nicht ausgeführt wird.

Um nach Fehlern bei der Überprüfung der Signatur durch den NASL Interpreter zu suchen könne die Option „-T“ verwenden und damit die Nachverfolgung („trace mode“) aktivieren. Die Überprüfung der Signatur wird auf diesem Wege ausführlichere Informationen bezüglich der Überprüfung in der „trace file“ aufzeichnen.

4 OpenVAS Dateien

(von Michael Wiegand)

Bei der Installation von OpenVAS werden Dateien an verschiedenen Orten in Ihrem Dateisystem abgelegt. Dieses Kapitel soll Ihnen einen groben Überblick geben, welche Dateien wohin installiert werden.

Bitte beachten Sie, dass die hier beschriebenen Orte sich auf eine Standardinstallation aus dem Quelltext mit `prefix=/usr/local` beziehen. Diese Orte können abweichen falls Sie bei der Konfiguration und Installation der einzelnen Modulen eine andere Auswahl getroffen haben; falls Sie OpenVAS aus einem von Ihrer Distribution zur Verfügung gestellten Binärpaket installiert haben, wurde eventuell andere Orte gemäß des Richtlinien Ihrer Distribution ausgewählt.

Beachten Sie auch, das viele der beschriebenen Verzeichnisse naturgemäß mehr Dateien beinhalten als die hier beschriebenen; diese Kapitel bezieht sich nur auf Dateien, die von OpenVAS an diesen Orten abgelegt wurden.

4.1 Ausführbare Dateien für Benutzer (PREFIX/bin)

Dieses Verzeichnis enthält ausführbare Dateien, die für den Benutzer oder während des Kompilierens hilfreich sein können.

- `libopenvas-config`: Ausgabe von Kompilierungs-Parametern für Module, die openvas-libraries nutzen
- `openvas-libnasl-config`: Ausgabe von Kompilierungs-Parametern für Module, die openvas-libnasl nutzen
- `openvasd-config`: Ausgabe von Kompilierungs-Parametern für Module, die openvas-server nutzen
- `openvas-mkcert-client`: Erstellung von Client-Zertifikaten
- `openvas-nasl`: Ein eigenständiger NASL-Interpreter
- `OpenVAS-Client`: Die grafische Benutzungsoberfläche

4.2 Server-Konfiguration (PREFIX/etc/openvas)

Dieses Verzeichnis enthält die systemweiten Konfigurationsdateien für OpenVAS, inklusive der Zugriffsregeln und der Benutzerdatenbank.

- `openvasd.conf`: Die zentrale Konfigurationsdatei des OpenVAS Servers wie in Kapitel 3.3.3 beschrieben.
- `openvasd.rules`: Regeln, die den Zugriff auf Zielsysteme für die Nutzer dieser OpenVAS-Installation einschränken wie in Kapitel 3.3.2 beschrieben.
- `gnupg`: Der Keyring und die Vertrauensgrade anhand derer festgelegt wird, welche NVTs vertrauenswürdig genug sind, um ausgeführt zu werden.

4.3 Datei für die Kompilierung (PREFIX/include/openvas)

Dieses Verzeichnis enthält die C-Headerdateien für OpenVAS. Diese Dateien sind nur für die Kompilierung von OpenVAS auf dem jeweiligen System relevant.

4.4 Bibliotheken (PREFIX/lib)

Dieses Verzeichnis enthält die statischen („.a“) und dynamischen („.so“) Programmbibliotheken, die von den Modulen openvas-libraries und openvas-libnasl genutzt werden.

4.5 NVTs (PREFIX/lib/openvas/plugins)

Dieses Verzeichnis enthält alle `.nasl`- und `.inc`-Dateien.

4.6 Ausführbare Dateien für OpenVAS-Server (PREFIX/sbin)

Dieses Verzeichnis enthält verschiedene ausführbare Dateien, die für die Administration von OpenVAS-Server nützlich sind:

- `openvasd`: Der OpenVAS Server.
- `openvas-adduser`: Programm zum Hinzufügen eines OpenVAS-Nutzers.
- `openvas-mkcert`: Programm zum Erstellen eines Serverzertifikates.
- `openvas-rmuser`: Programm zum Entfernen eines OpenVAS-Nutzers.
- `openvas-nvt-sync`: Programm zur Synchronisation mit NVT Feeds.

4.7 Bedienungsanleitungen für Benutzer (PREFIX/share/man)

Dieses Verzeichnis enthält die auf Unix-basierten Systemen gebräuchlichen Bedienungsanleitungen („man pages“) für die für OpenVAS-Anwender gedachten ausführbaren Dateien.

4.8 Manual pages für den Server (PREFIX/man)

Dieses Verzeichnis enthält die auf Unix-basierten Systemen gebräuchlichen Bedienungsanleitungen („man pages“) für die für OpenVAS-Administratoren gedachten ausführbaren Dateien.

4.9 Installationsspezifische Daten (PREFIX/var/lib/openvas)

Dieses Verzeichnis enthält Daten, die für die jeweilige OpenVAS-Installation spezifisch sind:

- `CA`: Öffentliche Zertifikate, die für den Aufbau einer verschlüsselten Verbindung erstellt wurden.

- `private/CA`: Die dazugehörigen geheimen Zertifikate.
- `users/`: Dateien für die einzelnen OpenVAS-Benutzer.
- `openvas-services`: Liste der bekannten Dienste und der zugehörigen Portnummern.
- `services.tcp`: TCP-Dienste.
- `services.udp`: UDP-Dienste.

4.10 Protokolldateien (PREFIX/var/log/openvas)

Dieses Verzeichnis enthält die Protokolldateien des OpenVAS-Servers:

- `openvasd.dump`
- `openvasd.messages`

4.11 Serverprozessinformation (PREFIX/var/run)

Dieses Verzeichnis enthält Laufzeitdaten einer in der Ausführung befindlichen OpenVAS-Installation:

- `openvasd.pid`: Datei zur Prozessidentifizierung.

4.12 Benutzerdaten (HOME/)

Alle benutzerspezifischen Daten, die von OpenVAS-Client verwaltet werden, sind im jeweiligen Benutzerverzeichnis („HOME“) abgelegt:

- `.openvas/TTT/`: Verzeichnis mit alle Dateien bezüglich Aufgabe „TTT“.
- `.openvas/TTT/nessusrc`: [OpenVAS 1.0] Aufgabenspezifische Konfiguration.
- `.openvas/TTT/openvasrc`: [OpenVAS 2.0] Aufgabenspezifische Konfiguration.
- `.openvas/TTT/SSS/`: Verzeichnis mit allen Dateien, die sich auf Bereich „SSS“ der Aufgabe „TTT“ beziehen.
- `.openvas/TTT/SSS/nessusrc`: [OpenVAS 1.0] Bereichsspezifische Konfiguration.
- `.openvas/TTT/SSS/openvasrc`: [OpenVAS 2.0] Bereichsspezifische Konfiguration.
- `.openvas/TTT/SSS/RRR/`: Verzeichnis mit allen Dateien, die sich auf den Bericht „RRR“ im Bereich „SSS“ der Aufgabe „TTT“ beziehen.
- `.openvas/TTT/SSS/RRR/nessusrc`: [OpenVAS 1.0] Die während der Erstellung des Berichtes „RRR“ gesetzte Konfiguration.
- `.openvas/TTT/SSS/RRR/openvasrc`: [OpenVAS 2.0] Die während der Erstellung des Berichtes „RRR“ gesetzte Konfiguration.
- `.openvas/TTT/SSS/RRR/report.nbe`: Der Inhalt des Berichtes „RRR“ in einem Textformat.
- `.openvas/TTT/SSS/RRR/report.nbe.cnt`: Zähler für die im jeweiligen Bericht enthaltenen Sicherheitsmeldungen. Dieser Zähler kann aus der Berichtsdatei report.nbe neu berechnet werden.

- `.openvas/TTT/SSS/RRR/nessus_plugin_cache`: [OpenVAS 1.0] Zwischenspeicher der Beschreibungen der NVTs, die bei der Erstellung des Berichtes „RRR“ eingesetzt wurden.
- `.openvas/TTT/SSS/RRR/openvas_nvt_cache`: [OpenVAS 2.0] Zwischenspeicher der Beschreibungen der NVTs, die bei der Erstellung des Berichtes „RRR“ 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.

5 OpenVAS-Client installieren und konfigurieren

(von Tim Brown, Jan-Oliver Wagner und Michael Wiegand)

5.1 Binärpakete installieren

Binärpakete für die verbreitetsten Linux-Distributionen und einige andere Unix-basierte Systeme sind über die OpenVAS-Website verfügbar.

Falls OpenVAS-Client noch nicht für das Betriebssystem oder die Distribution Ihrer Wahl verfügbar ist, ist es auf den meisten Unix-basierten Systemen in der Regel möglich, OpenVAS-Client selbst zu kompilieren (siehe Abschnitt 5.2). Bei Bedarf können Sie gerne die bei dieser Distribution für die Paketierung Verantwortlichen darauf hinweisen, dass Sie an OpenVAS-Paketen interessiert sind und dass das OpenVAS Entwickler-Team neue Paketierungen nach Möglichkeit unterstützt.

5.1.1 Debian und Ubuntu

OpenVAS-Client ist ein offizieller Bestandteil der Debian-Distributionen „unstable“ („Sid“) und „testing“ („Lenny“) und der Ubuntu-Distributionen ab Version 8.10 („Intrepid Ibex“). Das heißt, dass Sie OpenVAS-Client einfach mittels des `apt-get` Mechanismus installieren können, falls Sie eine dieser Distributionen verwenden. Bitte informieren Sie sich auf der OpenVAS-Website, falls Sie Binärpakete für ältere Versionen dieser Distributionen benötigen.

5.1.2 Gentoo

OpenVAS-Client Pakete für Gentoo sind über das „Portage“-System verfügbar. In der Gentoo-Dokumentation und auf der OpenVAS-Website finden Sie nähere Informationen zur Installation dieser Pakete.

5.1.3 RPM-basierte Distributionen

Pakete für einige Versionen von OpenSUSE, Fedora, Mandriva und anderen RPM-basierten Distributionen sind in inoffiziellen Paketquellen unter

http://download.opensuse.org/repositories/security:/openvas:/ und http://www.atomicorp.com/channels/atomic/

verfügbar. In der Dokumentation Ihrer Distributionen finden Sie eine Anleitung zum Hinzufügen von Paketquellen und zum Installieren dieser Pakete.

5.1.4 Windows XP SP2

Binärpakete für Microsoft Windows XP SP2 sind über die OpenVAS-Website verfügbar.

5.1.5 FreeBSD

Die FreeBSD Ports- und Paketesammlung enthält Ports und Pakete für OpenVAS-Client.

Mit dem folgenden Befehl können Sie den FreeBSD Port auf Ihrem FreeBSD System kompilieren und installieren:

```
cd /usr/ports/security/openvas-client/ && make install clean
```

Falls Sie das Binärpaket vorziehen, so können Sie dieses mit dem folgenden Befehl installieren:

```
pkg_add -r openvas-client
```

5.2 OpenVAS-Client kompilieren

5.2.1 Aktuelles Quelltext-Release

Auf der OpenVAS-Website finden Sie den Link zum aktuellen Quelltext-Release.

Wenn Sie ein Quelltext-Release nutzen möchten, sollten Sie bedenken, dass dabei eventuell im Laufe der Installation Dateien an andere Orte kopiert werden als die von Ihnen oder Ihrer Distribution erwarteten Orte im Dateisystem. Dadurch kann es durch Unachtsamkeit leicht zu unerwarteten Ergebnissen kommen, falls Sie versuchen, Releases einer anderen Serie zu installieren oder alte Versionen von OpenVAS-Client zu entfernen. Aus diesem Grund empfehlen wir Ihnen, die Module mit einem Präfix zu konfigurieren (bspw. `./configure prefix=/opt/openvas`) und für verschieden Serien unterschiedlich Präfixe zu benutzen. Auf diese Weise können Sie die zu einer bestimmten Version des OpenVAS-Clients gehörenden Dateien einfach isolieren; falls Sie eine bestimmte Installation entfernen möchten, können Sie einfach das jeweilige Unterverzeichnis löschen.

Laden Sie zunächst das Quelltextarchiv mit der Endung „.tar.gz“ herunter und entpacken Sie sie mit dem Befehl `tar -xzf openvas-client-N.N.N.tar.gz`. Das Kompilieren des Quelltextes ist zur Zeit auf GNU/Linux-Systeme ausgerichtet, funktioniert aber eventuell in anderen Umgebungen.

Bitte lesen Sie nun die Datei `README` in dem eben entpackten Verzeichnis, dort finden Sie weitere Anweisungen.

5.2.2 Aktueller Entwicklungsstand (direkt aus der Quelltextverwaltung)

Sie benötigen das Programm „subversion“, um den aktuellen Entwicklungsstand herunterzuladen:

```
 $ svn checkout
https://svn.wald.intevation.org/svn/openvas/trunk/openvas-client
```

Bitte lesen Sie nun die Datei `README` in dem neu erstellten Verzeichnis, dort finden Sie weitere Anweisungen.

Das Team der OpenVAS-Entwickler fühlt sich dazu verpflichtet, eine möglichst hohe Codequalität zu gewährleisten. Bitte beachten Sie trotzdem, dass Sie einen aktuellen Entwicklungsstand benutzen, der unvollständig und instabil sein könnte und nicht in Produktivumgebungen eingesetzt werden sollte.

6 OpenVAS-Client-benutzen

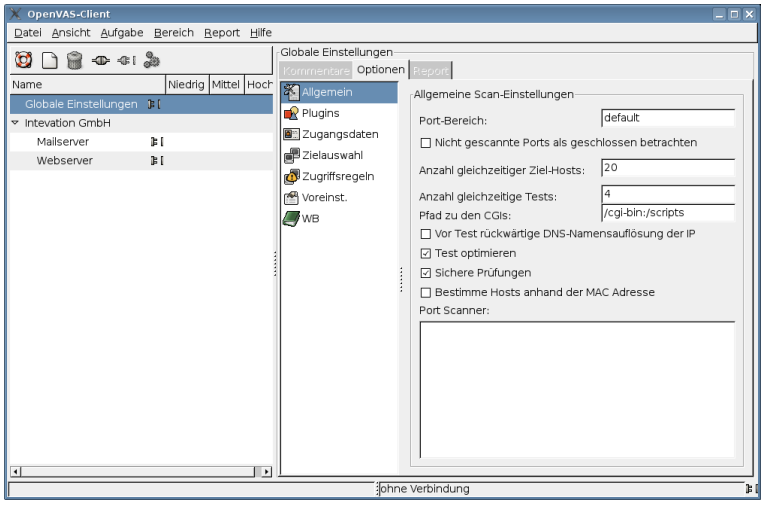
(von Jan-Oliver Wagner)

Dieser Abschnitt beschreibt den grundlegenden Aufbau von OpenVAS-Client und bietet eine Anleitung für die Verwendung von OpenVAS-Client in der alltäglichen Benutzung. Zusätzlich wird auf in diesem Abschnitt auf die besonderen Eigenschaften des Systems eingegangen, die besonders für fortgeschrittene Benutzer von OpenVAS interessant sein könnten.

Diese Dokumentation bezieht sich auf Version 2.0-beta1 von OpenVAS-Client. In neueren Versionen kann sich die Funktionalität geändert haben oder neue Funktionalität hinzugekommen sein. Falls Sie eine neuere Version nutzen, raten wir Ihnen, sich auf der OpenVAS Website über die Änderungen zu informieren und eine aktualisierte Version dieser Dokumentation herunterzuladen.

6.1 Das Hauptfenster

Das Hauptfenster von OpenVAS-Client teilt sich in zwei Bereiche: Links bietet eine Baumstruktur eine Übersicht lokal gespeicherter Aufgaben, Bereiche und Berichte. Im rechten Bereich finden sich Karteikarten mit Kommentaren, Einstellungen und Berichte. Hier lassen sich Scans konfigurieren und kommentieren und Ergebnisse sichten. Beachten Sie, dass die Karte der Einstellungen selbst wieder eine Sammlung von Karten enthält.



Nach dem ersten Start von OpenVAS-Client enthält die Baumstruktur nur einen Eintrag: Globale Einstellungen. Dies sind die Voreinstellungen für OpenVAS-Client. Diese berücksichtigen keine bestimmte Auswahl an Plugins, dafür ist eine Verbindung zu einem OpenVAS Server notwendig. Sie müssen eine Verbindung zu einem Server aufbauen und dann die Einstellungen für eine Auswahl von Plugins angeben. Speichern Sie dann Ihre bevorzugten Voreinstellungen über den Menüpunkt „Globale Einstellungen speichern“ des Menüs „Datei“.

6.1.1 Aufgaben

Der Sinn von Aufgaben ist es, alle Aktionen eines Oberthemas zusammen zu fassen. Oberthemen sind z.B. „Test aller Rechner am Standort ABC“ oder „Kunde XYZ GmbH“.

Einer Aufgabe kann ein Kommentar zugeordnet werden, der das Oberthema detaillierter beschreibt. Hier können auch weitere wichtige Informationen oder Hinweise angegeben werden: Etwa wann die nächste Serie von Scans durchgeführt werden soll oder auf welcher Vertragsgrundlage die Scans durchgeführt werden.

Eine Aufgabe hat keine Optionen oder Berichte. Sie stellt nur eine Sammlung von Bereichen dar.

Mögliche Aktionen bezüglich Aufgaben:

Neu Erzeugt eine neue Aufgabe mit dem Titel „unbenannte Aufgabe“.

Umbenennen Der Titel einer Aufgabe kann entweder direkt in der Baumstruktur durch Anklicken des Titels oder durch Auswahl des entsprechenden Menü-Eintrags geändert werden.

Löschen Löschen einer Aufgabe bedeutet das Löschen aller zugehörigen Bereiche! Daher erfolgt eine Sicherheitsabfrage vor der eigentlichen Löschung.

6.1.2 Bereiche

Ein Bereich kann als Teil-Aufgabe angesehen werden: Er bezeichnet einen bestimmten Sicherheits-Scan. Der Titel sollte aussagekräftig das Ziel des Scans beschreiben, z.B.: „Behutsamer Scan des produktiven Web-Servers“, „Aggressiver Scan des Web-Server Test-Systems“ oder „Alle Sun Workstations“.

Zu jedem Bereich kann ein Kommentar gespeichert werden, der die Bedeutung des Scans detaillierter beschreiben und sonstige hilfreiche Informationen enthalten kann.

Ein Bereich umfasst auch einen vollständigen Satz an Einstellungen für den Sicherheits-Scan. Wenn ein neuer Bereich erstellt wird, werden zunächst die „Allgemeinen Einstellungen“ kopiert. Die einzelnen Optionen werden weiter unten genauer beschrieben. Für jeden Bereich kann eine Verbindung zu einem OpenVAS Server aufgebaut werden. Anschließend kann die Auswahl von Plugins als Teil der Einstellungen durchgeführt werden. Ein Symbol rechts neben dem Titel zeigt die aktive Verbindung zu einem OpenVAS Server an. Sinn der Zuordnung eines Servers zu einem Bereich ist, dass jeder Server unterschiedliche Plugins anbieten kann. Mit dem Aufbau der Verbindung bezieht OpenVAS-Client eine Liste der verfügbaren Plugins.

Ein Bereich kann außerdem eine Sammlung von Berichten umfassen. Jedes Mal, wenn ein Scan erfolgreich durchgeführt wurde, wird ein neuer Bericht angelegt. Zusätzlich können Bericht aus Dateien geladen werden.

Bitte beachten Sie, dass Änderungen an einem Bereich werden immer und nur genau dann gespeichert werden, wenn ein Scan gestartet wird. Führen Sie beispielsweise Änderungen an der Plugin-Auswahl durch und schließen dann OpenVAS-Client, ohne einen Scan durchzuführen, werden die Änderungen verworfen.

Mögliche Aktionen bezüglich Bereiche:

Ausführen Die Konfiguration für den Bereich wird gespeichert und ein Sicherheits-Scan wird mit den aktuellen Einstellungen durch den verbundenen OpenVAS Server ausgeführt.

Neu Ein neuer Bereich mit dem Titel „unbenannter Bereich“ wird als Teil der aktuell ausgewählten Aufgabe angelegt. Als Voreinstellung werden die allgemeinen Einstellungen kopiert. Beachten Sie hier, dass nur explizit gespeicherte Globale Einstellungen als Voreinstellungen verwendet werden. Änderungen, die nur im OpenVAS-Client vorliegen, haben keine Auswirkung auf einen neuen Bereich.

Umbenennen Der Titel eines Bereichs kann entweder direkt in der Baumstruktur durch Anklicken des Titels oder durch Auswahl des entsprechenden Menü-Eintrags geändert werden.

Löschen Löschen eines Bereichs bedeutet das Löschen aller zugehörigen Berichte! Daher erfolgt eine Sicherheitsabfrage vor der eigentlichen Löschung.

Verschieben zu Aufgabe Ein Bereich mit allen zugeordneten Berichten kann von einer Aufgabe zu einer anderen verschoben werden. Ein Untermenü enthält die Liste aller Aufgaben, aus diesen kann das Ziel ausgewählt werden.

Öffnen Ein Bereich kann geladen und der aktuellen Aufgabe zugeordnet werden. Mit diesem Schritt werden jedoch nur die Einstellungen für den Bereich geladen! Die Berichte liegen in eigenen Dateien vor, die separat importiert werden können. Die Aktionen Öffnen und Speichern (siehe unten) erlauben es, Einstellungen für einen bestimmten Bereich an andere weiterzugeben oder selbst Sicherungskopien anzulegen.

Speichern als Der aktuellen Bereich wird in eine Datei gespeichert (in der Struktur von openvasrc). Es werden nur die Einstellungen des Bereichs gespeichert, nicht die zugehörigen Berichte.

6.1.3 Berichte

Ein Bericht ist das Ergebnis eines Sicherheits-Scans. Er enthält die Ergebnisse der ausgeführten Plugins zusammen mit Informationen zu Subnetz, Zielrechner, Port und Schwere des festgestellten Sicherheitsproblems.

In OpenVAS-Client besteht zusätzlich die Möglichkeit, für einen Bericht einen Kommentar anzugeben. Soweit verfügbar, werden auch die zum Scan gehörenden Einstellungen gespeichert. Diese zusätzlichen Informationen sind nicht Teil der Berichtsdateien und gehen daher verloren, wenn ein Bericht exportiert wird! Umgekehrt enthalten auch importierte Berichte keine Kommentare oder die zum Bericht gehörenden Einstellungen.

Mögliche Aktionen bezüglich Berichte:

Löschen Löscht den aktuellen Bericht und zugehörige Kommentare und Einstellungen. Es erfolgt eine Sicherheits-Abfrage vor der Löschung.

Import Importiert einen Bericht aus einer Datei. Der Standard für Austauschformate ist NBE (Dateiendung „.nbe“). Über den Dateiauswahl-Dialog kann der gewünschte Bericht ausgewählt werden. Konnte der Bericht nicht importiert werden, so erfolgt eine Fehlermeldung. Sonst wird der Bericht dem aktuellen Bereich hinzugefügt. Beachten Sie, dass weder Kommentare noch Einstellungen aus einer NBE-Datei importiert werden.

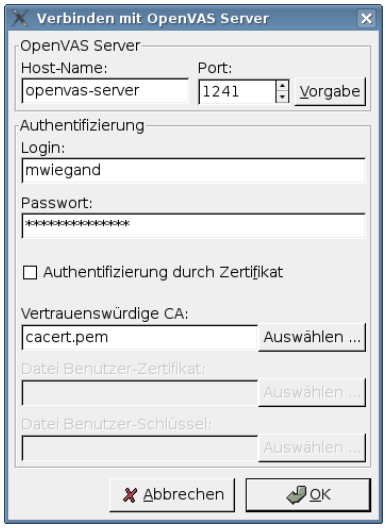
Export Der aktuell ausgewählte Bericht kann exportiert werden. Entweder in einem Nessus-Austauschformat (NBE) oder in einem Format zur Weiterverarbeitung bzw. Darstellung der Ergebnisse (XML, HTML, HTML mit Diagrammen, LaTeX, ASCII Text und PDF). Es wird empfohlen, das NBE-Format zu nutzen, wenn die Daten mit anderen ausgetauscht werden sollen und das PDF für einfache Berichte, die keiner weiteren Überarbeitung bedürfen. Die anderen Formate eignen sich zur Weiterverarbeitung der Ergebnisse bzw. zur Übernahme in eigene Dokumente.

Drucken Das Druck-Kommando erzeugt einen PDF-Bericht und startet einen auf dem System installierten PDF-Betrachter. Mittels dessen Druckfunktionen kann der Bericht dann gedruckt werden. Sollte trotz eines installierten Betrachters kein Programm gestartet werden, prüfen Sie bitte, ob sich die Installation in den aktuellen Suchpfaden befindet.

6.2 Authentifizierung

OpenVAS-Client benötigt eine Verbindung zu einem OpenVAS Server, um von diesem die Liste der verfügbarer Plugins zu erhalten und über diesem einen Sicherheits-Scan durchzuführen. Ab der Version 2.0 des OpenVAS-Clients wird der Nutzer nach einem erfolgreichen Verbindungsaufbau über neue Plugins informiert.

OpenVAS-Client kann verschiedene Verbindungen zu verschiedenen Servern gleichzeitig halten. Jedem Bereich ist eine eigene Verbindung zu einem Server zugeordnet. Zusätzlich kann für die allgemeinen Einstellungen ein Server verbunden sein, um die Voreinstellung für Plugin-Auswahl und -Parameter zu bestimmen. Beachten Sie hier, dass nur explizit gespeicherte Globale Einstellungen als Voreinstellung für neue Bereich verwendet werden.



Der Status der Verbindung zum Server wird durch ein Symbol neben dem Titel eines Bereichs oder den allgemeinen Einstellungen in der Baumstruktur dargestellt. Nur für Bereiche kann eine Verbindung zu OpenVAS Servern aufgebaut werden, nicht für Aufgaben oder Berichte.

Weitere Informationen zum Verbindungsstatus zeigt die Status-Zeile am Fuß des Hauptfensters: Hier werden die tatsächlichen Verbindungsdaten angezeigt, z.B. „Verbindung: username@host.test.example“.

Der „Verbinden“-Dialog bietet verschiedene Einstellungsmöglichkeiten für den Aufbau einer Verbindung zu einem OpenVAS Server:

Hostname Der Domain-Name oder die IP-Adresse des Rechners, auf dem der OpenVAS Server läuft.

Port Der Port, an dem der OpenVAS-Server auf Verbindungen wartet. Ältere Versionen des OpenVAS-Server einschließlich der Version 2.0.0 benutzten standardmäßig den Port 1241. Seitdem nutzt OpenVAS für die Kommunikation über das OpenVAS Transfer Protocol (OTP) den Port 9390. Mit dem „Vorgabe“-Knopf können Sie diese Einstellung jederzeit auf den Standardport zurücksetzen.

Login Ihr Nutzername auf dem ausgewählten Hostrechner. Um einen bestimmten OpenVAS Server benutzen zu können, benötigen Sie ein Nutzerkonto für diesen Server. Der Administrator des Servers kann ein Konto für Sie einrichten.

Passwort Das Passwort für Ihr Nutzerkonto auf einem OpenVAS Server.

SSL Optionen

Vertrauenswürdige CA: Das Zertifikat benennt eine Certificate Authority (CA), der Sie vertrauen. Mit diesem Zertifikat überprüfen Sie, ob Sie sich mit einem vertrauenswürdigen OpenVAS Server verbinden. Die Überprüfung wird für die „Paranoia Level“ 2 und 3 durchgeführt, nicht jedoch für den Level 1. Der Paranoia Level kann manuell in der Konfigurationsdatei .openvasrc eingestellt werden. Wenn Sie zum ersten Mal eine Verbindung zu einem OpenVAS Server aufbauen, werden Sie außerdem ausdrücklich nach dem gewünschten Paranoia Level gefragt.

Der vorgegebene Pfad für die vertrauenswürdige CA der Dateiname der lokalen OpenVAS Server Installation. Wenn Sie also OpenVAS-Client und -Server auf dem gleichen Rechner laufen lassen oder beide das gleiche Dateisystem nutzen, so können Sie diese Vorgabe einfach benutzen.

Wenn Sie sich mit einem entfernten OpenVAS Server verbinden, benötigen Sie eine Kopie des CA Zertifikats, die Sie an beliebiger Stelle in Ihrem Heimat-Verzeichnis ablegen können.

Authentifizierung durch Zertifikat: Wenn Sie diese Methode benutzen, benötigen Sie ein für Sie erstelltes Schlüssel/Zertifikat Paar. Dieses erstellt üblicherweise der Administrator des OpenVAS Servers mit den entsprechenden Skripten. Sie erhalten zwei Dateien: Ein Benutzer-Zertifikat und einen Benutzer-Schlüssel. Der Schlüssel kann mit einem Passwort versehen sein, dieses geben Sie dann bitte bei Verbindungsaufbau in das entsprechende Textfeld ein.

6.3 Scan Optionen

Dieser Abschnitt erläutert die wichtigsten Konfigurationsoptionen für einen Sicherheits-Scan (auf der Karte „Optionen“). Speziellere Optionen (z.B. Zugriffsregel, von der Wissensbasis gelöste Scans) werden später im Kapitel über spezielle Funktionen erläutert

6.3.1 Allgemein

Diese Karte enthält alle allgemeinen Scan-Einstellungen, vergleichen Sie dazu auch den Screenshot des Hauptfensters im Abschnitt 6.1.

Port-Bereich Die Auswahl der Ports, die durch den OpenVAS Server gescannt werden sollen. Es kann entweder ein Port-Bereich angegeben werden, wie etwa „1-8000“, oder komplexe Mengen wie „21,23,25,1024-2048,6000“. Die Auswahl „-1“ bedeutet, keine Portscans durchzuführen, „default“ bedeutet, die der OpenVAS Dienstedatei (openvas-services) angegebenen Ports zu scannen.

Nicht gescannte Ports als geschlossen betrachten Um Zeit zu sparen, können Sie dem OpenVAS Server vorgeben, TCP-Ports, die nicht gescannt wurden, als geschlossen zu betrachten. Die Überprüfung des Zielsystems ist damit unvollständig, aber die Scan-Zeit reduziert sich. Außerdem schickt der OpenVAS Server keine Pakete an Ports, die Sie nicht vorgegeben haben. Wenn diese Option nicht aktiviert ist betrachtet der OpenVAS Server Ports, deren Status unbekannt ist, als offen. Beachten Sie bitte, dass Sie eventuell Schwachstellen übersehen können, falls Sie diese Option aktivieren, etwa wenn auf Ihrem System Dienste auf Ports angeboten werden, die nicht zuvor gescannt wurden.

Anzahl Hosts, die zur gleichen Zeit getestet werden sollen Geben Sie hier die maximale Anzahl an Hosts an, die durch den Server gleichzeitig gescannt werden sollen. Beachten Sie, dass der OpenVAS Server für einen Scan max_hosts x max_tests Prozesse startet!

Anzahl Tests, die zur gleichen Zeit durchgeführt werden sollen Geben Sie hier die maximale Anzahl an Tests an, die durch den Server gleichzeitig gegen jeden Host gestartet werden. Beachten Sie, dass der OpenVAS Server für einen Scan max_hosts x max_tests Prozesse startet!

Pfad zu den CGIs OpenVAS bietet die Möglichkeit, mehrere Pfade („/cgi-bin“, „/cgis“, „/home-cgis“ usw.) nach CGIs zu durchsuchen. Geben Sie die zu durchsuchenden Pfade durch Doppelpunkte getrennt an: Etwa „/cgi-bin:/cgi-aws:/ openvas/cgi“.

Vor Test rückwärtige DNS-Namensauflösung der IP Ist diese Option eingeschaltet, so wird der OpenVAS Server eine DNS Rückwärts-Suche über die IP-Adresse durchführen (DNS reverse lookup) bevor die Tests durchgeführt werden. Dies kann die Durchführung der Tests insgesamt verlangsamen.

Test optimieren Sicherheits-Tests bitten unter Umständen den OpenVAS Server darum, nur dann gestartet zu werden, wenn bestimmte Informationen die durch andere Plugins gesammelt wurden, sich in der Wissen-Basis befinden oder nur wenn ein angegebener Port offen ist. Diese Option kann die Tests beschleunigen, kann aber dazu führen, dass der OpenVAS Server einige Lücken übersieht. Wer paranoid ist, schaltet diese Option aus.

Sichere Prüfungen Einige Sicherheitstests können dem Zielrechner Schaden zufügen, indem Dienste zeitweise oder bis zu einem Neustart dort nicht mehr zur Verfügung stehen. Wird diese Option eingeschaltet, so wird der OpenVAS Server sich auf gesendete Banner verlassen und keine echten Tests durchführen. Die Ergebnisse sind dann entsprechend weniger zuverlässig, aber wenigstens wird dadurch kein möglicher Ausfall für Benutzer entstehen. Unter Sicherheitsaspekten ist es angeraten, diese Option auszuschalten. Aus der Sicht eines Administrators sollte sie angeschaltet bleiben.

Bestimme Hosts anhand der MAC Adresse Wird diese Option eingeschaltet, so werden die Ziele im lokalen Netzwerk anhand ihrer Ethernet MAC Adresse anstelle der IP-Adresse bestimmt. Dies ist insbesondere dann sinnvoll, wenn OpenVAS in einem DHCP-Netzwerk verwendet wird. Sind Sie sich unsicher, dann schalten Sie die Option ab.

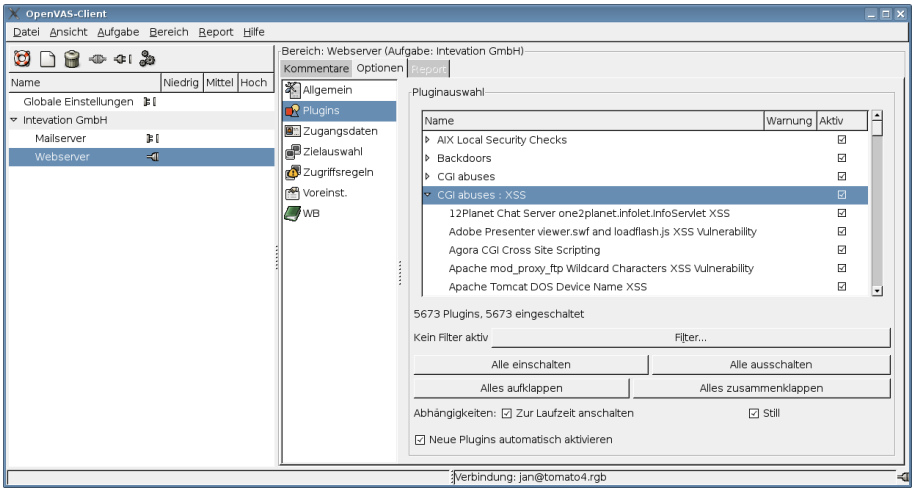
Portscanner Die Liste verfügbarer Portscanner. Portscanner stellen eine besondere Kategorie von Plugins dar und werden daher getrennt von den anderen Plugins dargestellt. Die Liste ist nur gefüllt, wenn eine Verbindung zu einem OpenVAS Server hergestellt wurde. Sie können einen oder mehrere Scanner auswählen und durch Anklicken mehr Details zu einem ausgewählten Scanner abrufen.

6.3.2 NVTs

Plugins werden auf dem OpenVAS Server gespeichert. Deshalb benötigen Sie eine Verbindung zu einem OpenVAS Server, um einen Auswahl zutreffen. Ansonsten wird eine leere Liste angezeigt.

Plugins werden unterschieden in verschiedene Familien, die als Ganzes durch entsprechende Checkboxen aktiviert oder deaktiviert werden können. Auch kann eine Familie aufgeklappt werden, so dass die zugehörigen Plugins aufgelistet werden und einzeln über die Check-Boxen aktiviert oder deaktiviert werden können um die Auswahl zu verfeinern.

Die Spalte „Warnung“ enthält ein Warnungs-Symbol für einige Plugins. Dieses Warnungs-Symbol bedeutet, dass das Plugin Dienste auf dem Zielsystem abschalten oder das gesamte System abschalten könnte. Sie sollten sehr vorsichtig sein, wenn Sie diese Option einschalten, da es nötig werden könnte, einige Dienste auf dem Zielsystem manuell neu zu starten.



Screenshot von OpenVAS-Client 2.0-beta1. Gezeigt wird die Ansicht zur Plugin-Auswahl.

Unterhalb der Plugin-Liste wird die Gesamtzahl der vom Server geladenen Plugins angezeigt. Desweiteren auch die Gesamtzahl der ausgewählten Plugins und die Anzahl der durch einen Filter dargestellten Plugins.

Die folgenden Aktionen sind möglich:

Alle einschalten Schaltet alle Plugin-Kategorien ein.

Alle ausschalten Schaltet alle Plugin-Kategorien aus.

Alles aufklappen Klappt die Plugin-Liste soweit auf, dass sämtliche Plugins angezeigt werden.

Alles zusammenklappen Es werden nur noch die Plugin Familien angezeigt.

Abhängigkeiten zur Laufzeit berücksichtigen Wird diese Option aktiviert, dann wird der OpenVAS Server alle Plugins einschalten, die von den bereits selektierten abhängig sind.

Stille Abhängigkeiten Wird diese Option aktiviert, dann wird der OpenVAS Server keinen Bericht für solche Plugins senden, die nicht explizit eingeschaltet wurden.

Filter Der Filter-Dialog erlaubt die Auswahl von Plugins anhand bestimmter Muster. Beachten Sie, dass Sie die vorherige Plugin-Auswahl komplett löschen sobald Sie einen Filter aktivieren.

Neue Plugins automatisch aktivieren Ab Version 2.0 des OpenVAS-Clients gibt es für den Nutzer die Möglichkeit auszuwählen, ob neue Plugins automatisch aktiviert werden sollen oder nicht. Direkt nach dem Verbindungsaufbau und dem Laden der neuen Plugins wird dem Nutzer ein Nachrichtenfenster angezeigt, das darüber informiert wieviele neue Plugins gefunden wurden und ob diese aktiviert wurden. Ältere Versionen des OpenVAS-Clients verhalten sich als ob diese Option angeschaltet wäre (aktivieren neue Plugins also automatisch), aber informieren den Nutzer nicht darüber, wieviele neue Plugins gefunden wurden.

Plugin Info-Dialog Wenn Sie auf einen Plugintitel einen Doppelklick ausführen, öffnet sich ein Dialog, der weitere Informationen zum Plugin bietet. Die angezeigten Informationen sind im Plugin abgelegt.

Folgende Aktionen sind im Info-Dialog möglich:

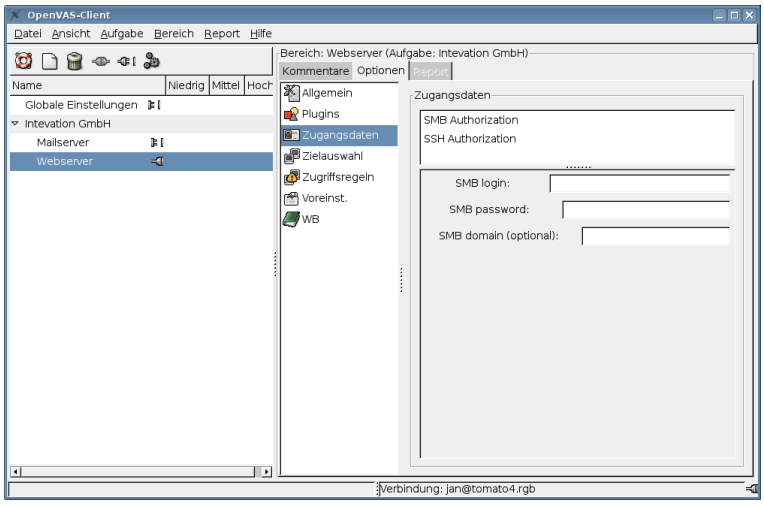
Setze Plugin Timeout Sie können eine Timeout für das Plugin angeben. Nach Ablauf dieser Zeitspanne wird die Ausführung des Plugins abgebrochen, falls das Plugin seine Ausführung noch nicht abgeschlossen hat.

Zeige Abhängigkeiten Ein weiterer Dialog listet die Abhängigkeiten des Plugins. Hier wird auch der Status (eingeschaltet/ausgeschaltet) der Plugins angegeben, von denen das aktuelle Plugin abhängt.

Informationen über Zertifikate Ab der OpenVAS-Client Version 2.0.0beta (welche auch eine entsprechende oder neuere Version des Servers benötigt) können die dem Server bekannten Zertifikate direkt vom Client aus angezeigt werden. Wenn das gerade im Plugin Info-Dialog angezeigte Plugin einfach oder mehrfach signiert wurde, werden hier Name und Vertrauensgrad der zu den Signaturen gehörenden Zertifikate aufgelistet. Daneben befinden sich Schaltflächen, um eine Ansicht des Zertifikats zu ermöglichen.

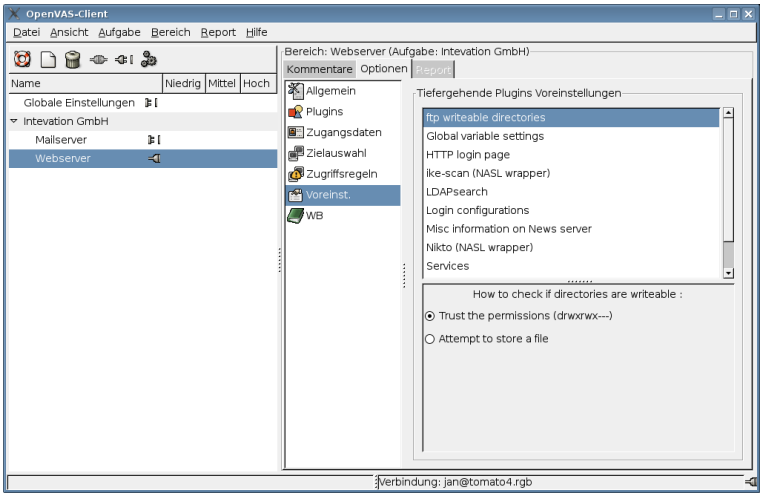
6.3.3 Zugangsdaten

Einige der Plugins bieten die Möglichkeit zur Eingabe von Zugangsdaten für bestimmte Anwendungen wie z.B. Samba oder für Web-Sites (HTTP). Diese Plugins funktionieren völlig analog zu den Plugins, die unter „Voreinstellungen“ eine Parametereingabe erlauben. Für eine bessere Übersicht ist die entsprechende Gruppe aber unter „Zugangsdaten“ getrennt aufgeführt.



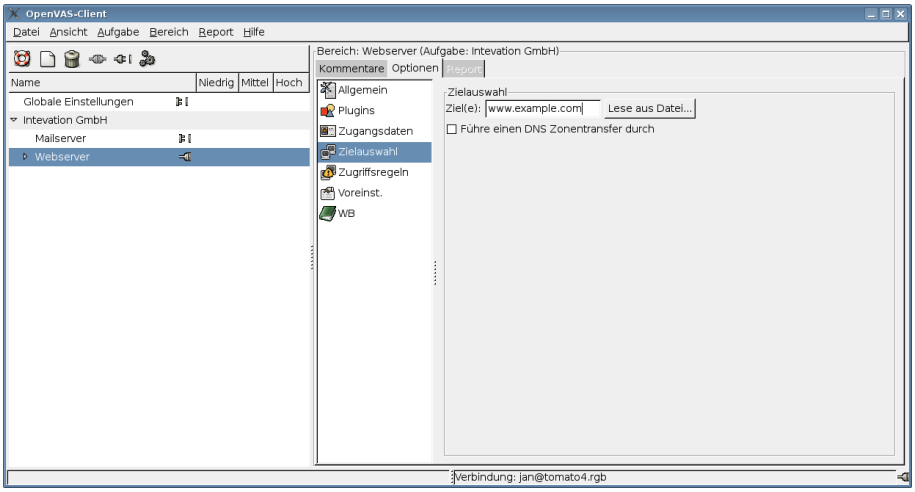
6.3.4 Pluginvoreinstellungen

Einige Plugins können anhand von Parametern angepasst werden. Die Parameter aller konfigurierbaren Plugins werden auf dieser Karte zusammengefasst und können modifiziert werden.



Nur eine vergleichsweise kleine Zahl von Plugins bietet die Möglichkeit einer Konfiguration.

6.3.5 Zielauswahl



Ziel(e) Die ersten Hosts, die durch den OpenVAS Server überprüft werden sollen. Die weiteren Optionen erlauben es, den Kreis der zu testenden Systeme zu erweitern. Es können verschiedene Ziele angegeben werden, indem sie als mit Komma (,) getrennte Liste angegeben werden, etwa „host1,host2“.

Eine besondere Syntax ist „file:/some/where/targetlist.txt“: Die Liste der Ziele wird aus einer Datei gelesen.

Lese aus Datei Es kann über den Dateiauswahl-Dialog eine Textdatei angegeben werden, welche eine Liste der Ziele enthält. Diese Textdatei kann eine oder mehrere Zeilen mit Komma-separierten Listen von Hosts enthalten.

Führe einen DNS Zonentransfer durch Der OpenVAS Server wird AXFR Anfragen (Zonen-Transfer) an den Nameserver des Ziels richten und so eine Liste der Hosts der Ziel-Domäne ermitteln. Dann wird jeder einzelne Host getestet.

6.3.6 Zugriffsregeln

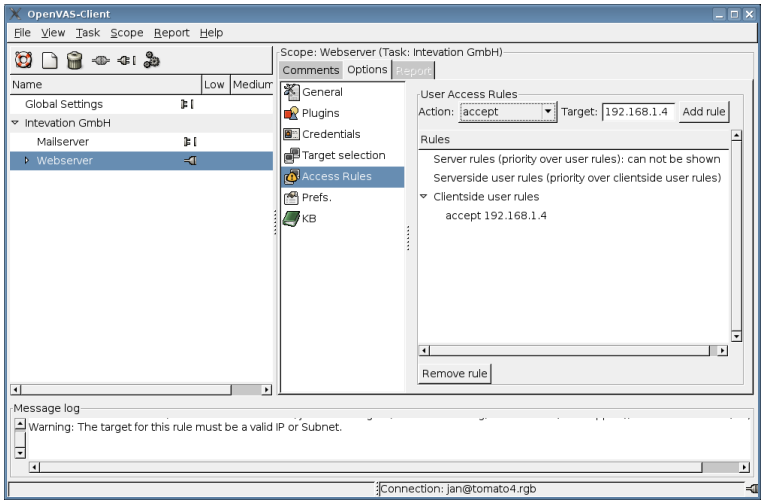
In diesem Abschnitt können Sie die Ihren Scan betreffenden Zugriffsregeln anschauen und verwalten. Diese Regeln bestimmen, welche Zielrechner Sie scannen dürfen. Beachten Sie, dass es drei verschiedene Arten von Regeln gibt.

Serverregeln Diese Regeln gelten für den gesamten Server und betreffen alle Benutzer, die sich mit diesem Server verbinden.

Server-seitige Benutzerregeln Diese Regeln gelten nur für einen bestimmten Benutzer und betreffen nur diesen Nutzer, egal von welchem Client aus er sich mit dem Server verbindet.

Klient-seitige Benutzerregeln Diese Regeln gelten für den jeweiligen Client. Sie betreffen nur den Bereich, in dem sie definiert sind.

Die ersten zwei Regelsätze sendet der Server nur zu Informationzwecken an den Client. Diese Regelsätze sind serverseitig und können nicht vom Client verändert werden. Nur der letzte Regelsatz kann vom Client geändert werden.

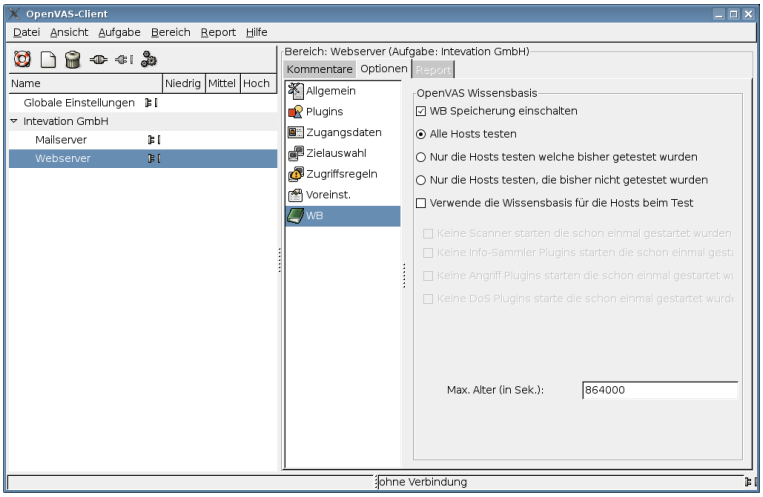


Eine Regel besteht aus einer Aktion und einem Ziel. Um eine Regel zu den clientseitigen Benutzerregeln hinzuzufügen, wählen Sie die gewünschte Aktion aus, geben Sie die IP-Adresse des Zielrechners oder ein Netzmaske ein und drücken Sie die Schaltfläche „Hinzufügen“. Sie können clientseitige Benutzerregeln entfernen, indem Sie sie auswählen und die Schaltfläche „Entferne Regel“ betätigen.

Weitere Informationen über Regeln und die Syntax der Regelsätze finden Sie in Abschnitt 3.3.2, der die Definition der serverseitigen Regeln erläutert.

6.3.7 Wissensbasis

Der Konfigurationsabschnitt für die Wissensbasis („Knowledge Base“, KB) erlaubt es Ihnen, die Verwaltung der serverseitigen Scanergebnisse zu steuern. Informationen, die während des Scans eines Zielrechners gesammelt werden, werden auf dem Service in einer Wissensbasis abgelegt. Es wird dabei pro Zielrechner einen Wissensbasis erstellt. Im Allgemeinen wird die jeweilige Wissensbasis automatisch gelöscht, sobald der Scan des Zielrechners abgeschlossen ist; unter gewissen Umständen kann es aber durchaus sinnvoll sein, die Wissensbasis aufzubewahren und sie zu einem späteren Zeitpunkt weiter zu nutzen.



Die folgenden Optionen stehen Ihnen für die Handhabung der Wissensbasis zur Verfügung:

WB Speicherung einschalten Wenn der Server nach dem Scan die Wissensbasis aufbewahren soll, müssen Sie diese Option aktivieren.

Alle Hosts testen Wenn diese Option aktiviert ist, wird des Server nicht die Wissensbasis zur Auswahl der zu scannenden Zielrechner verwenden, sondern alle vorgegebenen Zielrechner scannen.

Nur die Hosts testen welche bisher getestet wurden Falls das Speichern der Wissensbasis aktiviert ist, wird für jeden Zielrechner, der gescannt wurde, eine Wissensbasis angelegt. Dies kann dazu verwendet werden, nur Zielrechner zu überprüfen, die bereits in der Vergangenheit überprüft wurden. Das kann nützlich sein, wenn Sie eine gewisse Anzahl von Rechnern und die von ihnen angebotenen Dienste im Auge behalten wollen. Berücksichtigen Sie aber, dass diese Konfiguration dazu führen kann, dass Sie neue Rechner im Netzwerk nicht bemerken, da der Server sie nicht scannen wird.

Nur die Hosts testen, die bisher nicht getestet wurden Eine andere Möglichkeit das Vorhandensein einer Wissensbasis zu nutzen ist, alle Zielrechner auszuschließen, die bereits in der Vergangenheit gescannt wurden. Auf diese Weise findet ein neuer Scan automatisch Rechner, die seit dem letzten Scan zum Netzwerk hinzugekommen sind. Berücksichtigen Sie, dass dies dazu führt, dass Zielrechner nur jeweils einmal überprüft werden (wenn sie das erste Mal im Netzwerk gefunden werden). Sicherheitslücken, die sich seit diesem Scan entwickelt haben oder die nur von aktuellen NVTs gefunden werden, könnten Ihnen auf diese Weise entgehen.

Verwende die Wissensbasis für die Hosts beim Test Diese Einstellung steuert, ob der OpenVAS Server die beim letzten Scan erstellte Wissensbasis für den nächsten Scan wiederherstellen soll. Im Normalfall wird jedesmal eine neue Wissensbasis erstellt, wenn ein Rechner gescannt wird eine eventuell existierende Wissensbasis mit den Ergebnissen des aktuellen Scans ersetzt.

Keine Scanner starten die schon einmal gestartet wurden Falls der Server dazu konfiguriert wurde, eine existierende Wissensbasis erneut zu verwenden, werden durch die Aktivierung dieser Einstellung Portscanner nicht erneut ausgeführt, wenn von ihnen bereits ein Ergebnis in der Wissensbasis vorliegt.

Keine Info-Sammler starten die schon einmal gestartet wurden Falls der Server dazu konfiguriert wurde, eine existierende Wissensbasis erneut zu verwenden, werden durch die Aktivierung dieser Einstellung NVTs, die Informationen sammeln, nicht erneut ausgeführt, wenn von ihnen bereits ein Ergebnis in der Wissensbasis vorliegt.

Keine Angriff Plugins starten die schon einmal gestartet wurden Falls der Server dazu konfiguriert wurde, eine existierende Wissensbasis erneut zu verwenden, werden durch die Aktivierung dieser Einstellung Angriffs-NVTs nicht erneut ausgeführt, wenn von ihnen bereits ein Ergebnis in der Wissensbasis vorliegt.

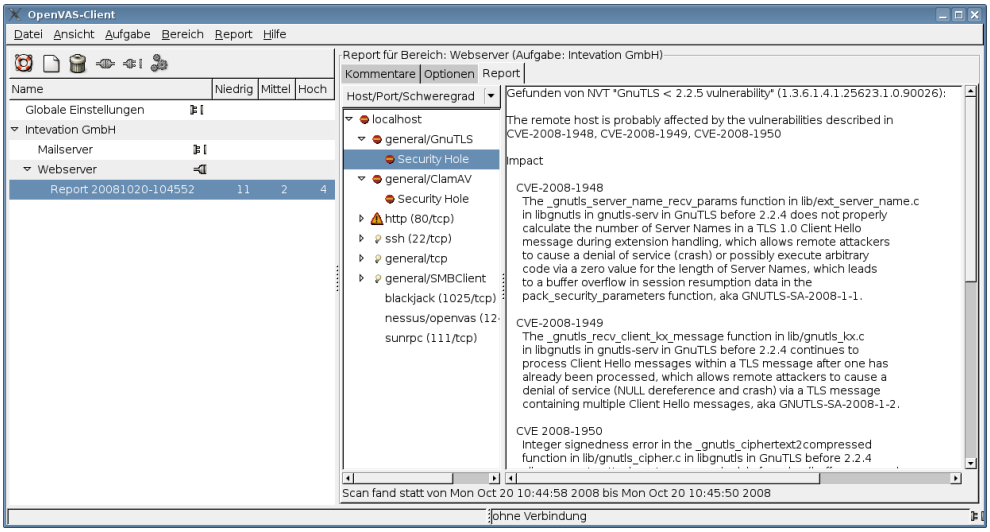
Keine DoS Plugins starten die schon einmal gestartet wurden Falls der Server dazu konfiguriert wurde, eine existierende Wissensbasis erneut zu verwenden, werden durch die Aktivierung dieser Einstellung „Denial of Service“-NVTs nicht erneut ausgeführt, wenn von ihnen bereits ein Ergebnis in der Wissensbasis vorliegt.

Max. Alter Diese Einstellung steuert das maximale Alter einer Wissensbasis (in Sekunden). Eine Wissensbasis, die älter als dieser Wert ist, wird automatisch verworfen.

6.4 Berichte

6.4.1 Berichte-Seite von OpenVAS-Client

Die Berichte-Seite enthält drei Elemente: Auf der linken Seite befindet sich eine Baumstruktur, welches es erlaubt über Host, Port und Schweregrad einzelne Scanergebnisse zu finden. Oberhalb dieser Baumstruktur ist eine Auswahl mit der man Baumstruktur umstellen kann. Auf der rechten Seite befindet sich das Textfeld mit den eigentlichen Scanergebnissen. Das gesamte Design unterstützt ein exploratives Verstehen der Scanergebnisse.



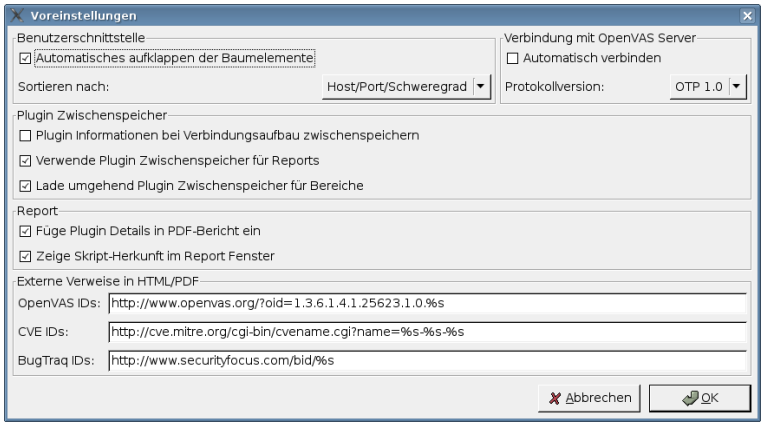
6.4.2 Berichtformate

Die Scan-Ergebnisse können in verschiedene Formate exportiert werden. Grundsätzlich kann zwischen drei Formaten unterschieden werden: Austauschformate, bearbeitbare Dokumente und Read-only Dokumente. Der letzte Typ ist aktuell der PDF Report. Mit einem geeigneten Betrachter können Sie im Dokument anhand der eingebetteten Hyperlinks navigieren.

Weitere Informationen zu den Formaten finden Sie im Abschnitt 6.1.3 über die Aktion „Report->Export“.

6.5 OpenVAS-Client Voreinstellungen

OpenVAS-Client erlaubt es mittels Voreinstellungen das Verhalten der graphischen Benutzungsoberfläche anzupassen.



Folgende Einstellungen sind verfügbar:

6.5.1 Benutzerschnittstelle

Automatisches Aufklappen der Baumelemente In der linksseitigen Baumansicht werden die Teilbäume für Aufgaben oder Bereiche jeweils automatisch aufgeklappt, wenn Sie sie anwählen und diese Voreinstellung eingeschaltet ist.

Ist die Option nicht ausgewählt, so wird ein Teilbaum erst dann aufgeklappt, wenn Sie direkt auf das entsprechende Symbol klicken.

Ordnen nach Diese Einstellung regelt die Sortierung der Scanergebnisse im Berichtsfenster. Standardmäßig werden die Ergebnisse zuerst nach Rechner, dann nach dem Port und schließlich nach dem Schweregrad sortiert. Unter gewissen Umständen kann die zweite Sortierungsvariante – die Sortierung nach der Portnummer – für Sie sinnvoller sein; etwa wenn Sie sich einen Überblick verschaffen wollen, welche Rechner in Ihrem Netzwerk einen möglicherweise anfälligen Dienst anbieten.

6.5.2 Verbindung mit dem OpenVAS Server

Automatisch verbinden Ist diese Option eingeschaltet, wird OpenVAS-Client versuchen eine Verbindung zum Server aufzubauen, sobald ein Bereich ausgeführt werden soll. Bei Verwendung von Benutzer-Zertifikaten ohne Passwort funktioniert dies unmittelbar. Für passwortgeschützte Benutzerzertifikate oder einfache passwortbasierte Anmeldung wird das Passwort im Speicher gehalten bis OpenVAS-Client beendet wird.

Protokollversion Diese Einstellung steuert, welches Protokoll der Client für die Kommunikation zwischen Server und Client anfordert. Beachten Sie bitte, dass der Server die Verbindung beenden wird, falls der Client ein nicht unterstütztes Protokoll anfordert.

6.6 Plugin Zwischenspeicher

Plugin Informationen bei Verbindungsaufbau zwischenspeichern Ist dieses Option eingeschaltet, legt OpenVAS-Client für den entsprechenden Bereich einen Zwischenspeicher für die kompletten Plugin Informationen an. Das wirkt sich auf dreierlei Weise aus:

Erstens kann der erneute Verbindungsaufbau zum OpenVAS Server bedeutend schneller sein, da Prüfsummen verwendet werden, um veränderte und neue Plugins zu identifizieren. Nur diese Änderungen werden dann heruntergeladen.

Zweitens sind sämtliche Plugin Informationen auch dann in OpenVAS-Client verfügbar wenn man noch keine Verbindung zum OpenVAS Server aufgebaut hat. Sie können sich also ohne Server-Verbindung die Pluginauswahl und die Plugin Voreinstellungen anschauen oder ändern. Beachten Sie, dass sich die Pluginauswahl ggf. beim Verbindungsaufbau ändern kann, beispielsweise wenn neue Plugins hinzugekommen sind oder gelöscht werden. Den Zwischenspeicher zu laden kann unter Umständen einige Sekunden dauern. Wollen Sie das vermeiden, dann schalten Sie die Option „Lade umgehend Plugin Zwischenspeicher für Bereiche“ aus.

Drittens der Nachteil eines Zwischenspeichers: Es werden pro Bereich einige MByte an Platz benötigt. Stellt dies ein Problem dar, sollten Sie diese Option abschalten. Wollen Sie bereits erzeugte Zwischenspeicher wieder entfernen, so suchen Sie nach den Datei `openvas_nvt_cache` in Ihrem OpenVAS Verzeichnis „ /.openvas“. Einfaches Löschen dieser Dateien reicht aus.

Verwende Plugin Zwischenspeicher für Berichte Falls Sie diese Option einschalten, wird OpenVAS-Client sämtliche Plugin Informationen allen neuen Scanberichten beilegen. Dies erlaubt es Ihnen, die Pluginauswahl und die Voreinstellungen der Plugins für einen Bericht in der grafischen Benutzungsoberfläche anzuschauen. Dieser Zwischenspeicher dient also der Verbesserung der Transparenz, nicht der Performanz.

Auch hier gibt es den Nachteil, dass pro Bericht mehrere MByte an Platz benötigt werden. Stellt dies ein Problem dar, sollten Sie diese Option abschalten. Wollen Sie bereits erzeugte Zwischenspeicher wieder entfernen, so suchen Sie nach den Datei `openvas_nvt_cache` in Ihrem OpenVAS Verzeichnis „ /.openvas“. Einfaches Löschen dieser Dateien reicht aus.

Lade umgehend Plugin Zwischenspeicher für Bereiche Falls Sie diese Option ausschalten, wird OpenVAS-Client den Zwischenspeicher für einen Bereich nicht automatisch laden, wenn Sie einen Bereich anwählen. Das bedeutet, dass Sie weder die Pluginauswahl noch die Plugin Einstellungen zu sehen bekommen, wenn Sie nicht mit dem OpenVAS Server verbunden sind. Diese Option schaltet also den zweiten Punkt von „Plugin Informationen bei Verbindungsaufbau zwischenspeichern“ wieder ab, um möglicherweise störende längere Ladezeiten des Zwischenspeichers beim Anwählen eines Bereichs zu vermeiden.

6.6.1 Bericht

Füge Plugin Details in PDF-Bericht ein Falls Sie diese Option eingeschaltet wird OpenVAS-Client bei der Erstellung von PDF-Berichten einen Anhang mit Details zu den Plugins anfügen, die die für diesen Bericht relevante Ergebnisse erzeugt haben. Diese Details sind innerhalb des PDF-Dokumentes verknüpft, so dass Sie leicht zu den gewünschten Informationen springen können.

Bedenken Sie aber, dass das PDF-Dokument dadurch unter Umständen signifikant an Umfang gewinnen kann. Im Durchschnitt können Sie davon ausgehen, dass etwa zwei Plugin-Beschreibungen auf einer Seite stehen.

Externe Verweise in HTML/PDF Diese Einstellungen bestimmen die URL für die Verweise auf weitere Informationen zu OpenVAS Plugins, CVE/CAN und BugTraq ID in Berichten der Formate HTML und PDF. Die Voreinstellungen wie oben im Screenshot zu sehen werden empfohlen, da dort jeweils aktuelle Informationen vorhanden sind. Die Voreinstellungen erhält man zurück wenn man die entsprechenden Felder leer lässt.

Für den Fall, dass Sie eine Offline-Version eines OpenVAS-Berichts als Paket mit etwa den Details zu CVE/CAN erstellen wollen, können Sie diese Definition verwenden: „mitre/%s/%s/%s.html“, falls Sie eine Verzeichnisstruktur relativ zur Berichtdatei haben, in der die entsprechenden HTML-Datei unter mitre/CVE/yyyy/nnnn.html

und mitre/CAN/yyyy/nnnn.html abgelegt sind (wobei yyyy für das Jahr und nnnn für die Nummer des Eintrages steht). Nun können Sie alle Dateien in diesem Verzeichnis in einem Paket zusammenfassen und versenden.

Beachten Sie, dass die Zeichenketten, die Sie hier definieren, in den Parameter „href“ der HTML-Verweise direkt eingetragen werden. Da Werkzeug „htmldoc“ wird verwendet um PDF-Berichte daraus herzustellen. Abhängig davon, welche Version mit welchen Eigenschaften Sie auf Ihrem System haben, kann die Art der erzeugten Verweise in der PDF-Datei variieren.

6.6.2 Installation von SLAD mit SLADinstaller

Falls Sie SLAD („Security Local Auditing Daemon“) wie in Abschnitt 8.1.1 beschrieben mit OpenVAS einsetzen wollen, bietet OpenVAS eine einfache Methode zur Installation von SLAD auf einem lokalen oder entfernen Rechner. Über den Menüpunkt „SLAD installieren“ im Dateimenü können Sie das Programm `sladinstaller` starten, welches Sie durch den weiteren Installationsablauf führt.

Bitte beachten Sie, dass das Programm `sladinstaller` in Ihrem Systempfad verfügbar sein muss, damit Sie diese Methode nutzen können. Weitere Informationen zur Installation und Benutzung von `sladinstaller` finden Sie in Abschnitt 8.1.1.

7 Durchführung von Local Security Checks

(von Jan-Oliver Wagner)

7.1 Debian Local Security Checks

Dieser Abschnitt beschreibt, wie Sie lokale Sicherheitstests („Local Security Checks“) mit OpenVAS durchführen können. Der hier beschriebene Ablauf wurde bis jetzt nur für lokale Sicherheitstests unter Debian getestet.

7.1.1 Voraussetzungen

Um lokale Sicherheitstests durchführen zu können, benötigen Sie eine funktionstüchtigen OpenVAS-Server. Informationen zur Installation und Konfiguration von OpenVAS-Server finden Sie in Kapitel 3 ab Seite 17.

7.1.2 Benutzer für Local Security Checks erstellen

Als erstes benötigen Sie einen Schlüssel mit Zertifikat:

```
$ ssh-keygen -t rsa -f ~/.ssh/id_rsa_sshovas -C
"OpenVAS-Local-Security-Checks-Key"
$ openssl pkcs8 -topk8 -v2 des3 -in ~/.ssh/id_rsa_sshovas -out sshovas_rsa.p8
```

Beachten Sie, dass der Kommentar (hier: „OpenVAS-Local-Security-Checks-Key“) keine Leerzeichen enthalten darf. Zur Zeit benötigen Sie einen Schlüssel vom Typ „RSA PKCS8“ für lokale Sicherheitstests aus OpenVAS.

Führen Sie nun auf jedem Zielsystem die folgenden Befehle durch:

```
# adduser --disabled-password sshovas
Name: OpenVAS Local Security Checks
# su - sshovas
$ mkdir .ssh
$ cp /some/path/id_rsa_sshovas.pub .ssh/authorized_keys
$ chmod 500 .ssh
$ chmod 400 .ssh/authorized_keys
```

7.1.3 Local Security Checks in OpenVAS-Client konfigurieren

Konfigurieren Sie in den Voreinstellungen die SSH-Anmeldung mit dem Schlüssel und dem Zertifikat, das Sie soeben erzeugt haben:

```
SSH login name: sshovas
SSH private key: ~/.ssh/sshovas_rsa.p8
SSH key passphrase: ********
SSH public key: ssh/id_rsa_sshovas.pub
```

Beachten Sie: Es ist im Allgemeinen nicht notwendig, den öffentlichen Schlüssel zu übertragen. Bedingt durch einen Fehler in dem von Nessus übernommenen Code muss dies aber zur Zeit durchgeführt werden.

Stellen Sie als nächstes sicher, dass Sie mindestens die folgenden NVTs aktiviert haben:

- Debian Local Security Checks/*
- Misc/Determine List of installed packages via SSH login
- Service Detection/Services
- Settings/Global variable settings
- Settings/SSH Authorization

oder stellen Sie sicher, dass Abhängigkeiten zur Laufzeit aufgelöst werden, falls Sie nicht alle lokalen Sicherheitstests ausführen möchten.

7.2 Windows Local Security Checks

Um – analog zu den lokalen Sicherheitstests für Linux – ein Gerüst für die lokale Suche nach Sicherheitslücken in Microsoft Windows-Systemen zur Verfügung zu stellen, hat das Team der OpenVAS-Entwickler eine neue Programmierschnittstelle („Application Programming Interface“, API) eingeführt.

Diese API (für die „Nessus Attack Scripting Language“, NASL) ermöglicht es, sowohl Binärdateien (wie etwas DLLs) als auch auf Metadaten des Microsoft Windows-Systems zu überprüfen, indem dafür die voreingestellten administrativen Freigaben des Windows-Systems genutzt werden.

Die Funktionalität ist ähnlich zu den „Nessus Windows Local Security checks“. Im Gegensatz zu dieser Implementierung benutzt OpenVAS für den Zugriff `samba` (`smbclient`) und implementiert das binäre SMB-Protokoll nicht selbst.

Der Vorteil dieser Integration von `smbclient` ist, dass auf diese Weise flexibler und schneller auf Änderungen des SAMBA/CEFIS-Protokolls reagiert werden kann.

7.2.1 Vorbereiten des OpenVAS Servers

Um die lokalen Sicherheitstests für Windows zu installieren müssen zunächst einige Schritte auf dem Rechner vorgenommen werden, auf dem der OpenVAS-Serverdienst (`openvasd`) läuft.

Die genaue Durchführung dieser Schritte hängt von dem Betriebssystem dieses Rechners ab; genauere Informationen finden Sie in der mit Ihrem System ausgelieferten Dokumentation.

1. SAMBA installieren (http://www.samba.org) Installationspakete sollten in der Regel für Ihr Betriebssystem verfügbar sein.
2. Stellen Sie sicher, dass die ausführbare Datei `smbclient` in Ihrem Systempfad gefunden werden kann.
3. Überprüfen die Einstellungen zu den Rechten bei den ausführbaren Dateien `openvasd` und `smbclient`: `smbclient` muss mit den Rechten von `openvasd` ausführbar sein.

7.2.2 Vorbereiten des Microsoft Windows Zielrechners

Die Implementierung der lokalen Sicherheitstests für Microsoft Windows wurde auf den folgenden Betriebssystemen getestet:

- Windows NT 4.0
- Windows 2000
- Windows XP SP2
- Windows XP SP3
- Windows Vista

Wahrscheinlich sind die lokalen Sicherheitstests für Microsoft Windows auch mit anderen Versionen kompatibel. Sofern der SMB-Port des Zielrechners erreichbar ist, kann die Version des Betriebssystems und die SAMBA-Version ermittelt werden. Für eine weiterführende Überprüfung sind die folgenden Schritte notwendig:

Sie benötigen administrative Zugriffsrechte auf den Zielrechner. Diese erhalten Sie in der Regel mit dem Benutzernamen (Voreinstellung: „Administrator“) und dem Passwort dieses Benutzers. Es gibt kein voreingestelltes Passwort, dass Passwort wurde während der Installation des Betriebssystems festgelegt.

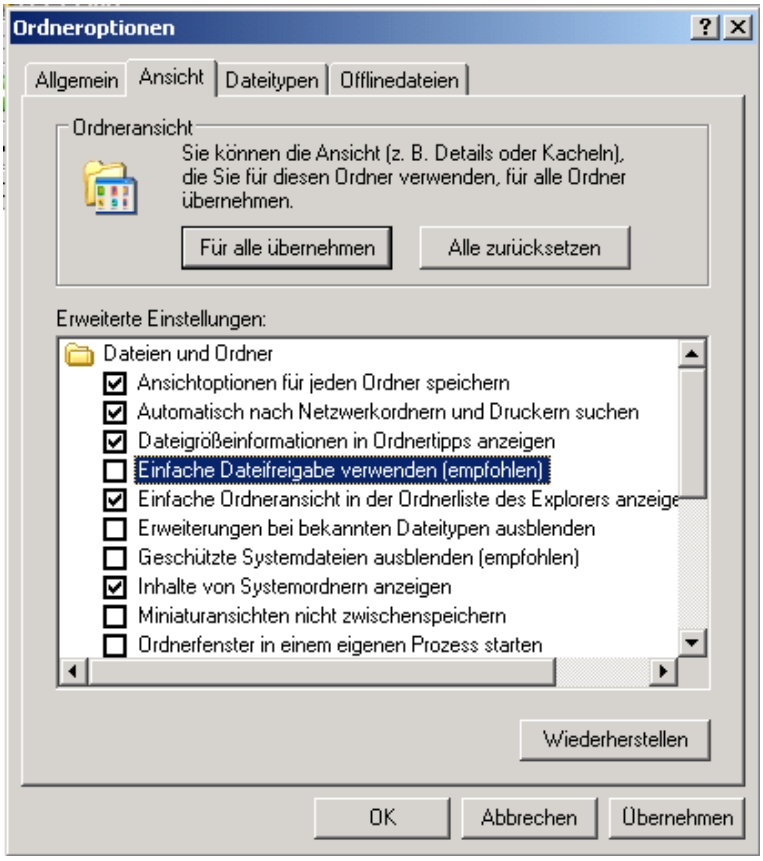
Die Zugriffsinformationen werden in der grafischen Benutzungsoberfläche von OpenVAS-Client als „SMB Credentials“ angegeben; diese Informationen werden für jeden Rechnern in der Liste der Zielsysteme verwendet.

Falls Sie planen, eine ganze Windows-Domäne zu scannen, können Sie an dieser den Benutzernamen und das Passwort des Domänen-Administrators angeben.

Stellen Sie sicher, dass während des Scans keine Firewall auf den Zielsystem aktiv ist oder dass eine entsprechende Ausnahmeregel für den OpenVAS Server vorhanden ist.

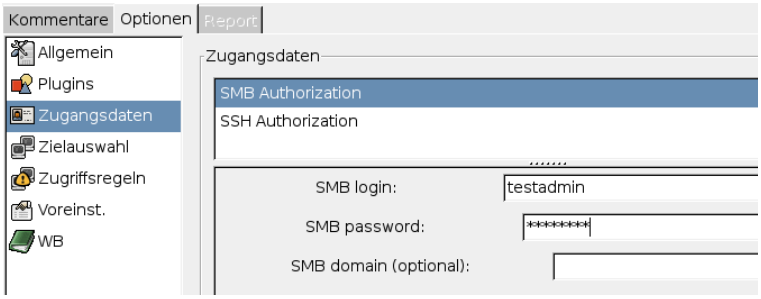
Anmerkung für Windows XP Bei der Verwendung von Windows XP ist es wichtig, dass die Einstellungen „Einfache Dateifreigabe verwenden“ nicht aktiviert ist.

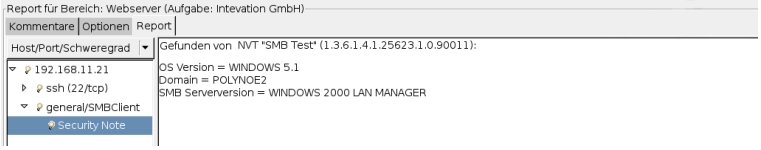
Ohne diese Einstellung ist `smbclient` nicht in der Lage, auf Dateien in den Windows-Systemfreigaben zuzugreifen.



7.2.3 Ausführung der Tests in OpenVAS-Client

Nachdem Sie in OpenVAS-Client den Zielrechner und die Zugriffsinformationen angegeben und den Scan ausgeführt haben, werden Informationen über das Zielsystem im Bericht erscheinen:





8 Integrierte Sicherheitswerkzeuge verwenden

8.1 Security Local Auditing Daemon (SLAD)

(von Jan-Oliver Wagner)

8.1.1 Security Local Auditing Daemon (SLAD) mit OpenVAS benutzen

Für die Installation von SLAD auf entfernten Systemen steht Ihnen mit `sladinstaller` ein Programm zur Verfügung, das automatisch die aktuelle Version bereitstellt und installiert.

Die Einbindung von `sladinstaller` in OpenVAS wird zur Zeit überarbeitet; auf der OpenVAS-Website finden Sie aktuelle Informationen zur Installation von `sladinstaller`.

8.1.2 SLAD Plugins

Bei SLAD handelt es sich um einen Dienst, der anderer Programme einbindet und eine einheitliche Schnittstelle zu ihren Ausgaben zur Verfügung stellt. In der aktuellen Version enthält SLAD die folgenden Plugins:

chkrootkit

`chkrootkit` ist ein Programm, das das lokale System auf Anzeichen für installierte Rootkits untersucht.

clamav

Das `clamav` stellt einen GPL-lizenzierten Virenscanner für Linux zur Verfügung. Es lässt sich steuern, ob Archive („.zip“, „.tar.gz“ usw.) gescannt werden sollen oder nicht und ob infizierte Dateien entfernt werden sollen oder in einem Quarantäne-Bereich isoliert werden sollen.

john

„John the ripper“ ist ein schneller Passwortbrecher. Ziel dieses Programmes ist es, schwache Benutzerpasswörter zu finden, die die Sicherheit des Systems beeinträchtigen könnten. Drei Optionen sind verfügbar:

Fast crack mode: In diesem Modus testet John lediglich den Benutzernamen und davon abgeleitete Wörter gegen die Hashwerte 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.

Die Standardversion von John zeigt gebrochene Passwörter im Klartext an. Dies macht es schwierig, John in einer professionellen Umgebung einzusetzen. Aus diesem Grund nutzt SLAD eine Version von John the Ripper, die dahingehend verändert wurde, dass nicht die Passwörter, sondern die Namen der Benutzer mit schwachen Passwörtern angezeigt werden.

lsof

Das Unix-Systemprogramm `lsof` zeigt eine Liste der zur Zeit auf dem System offenen Dateien an und welche Programme diese Dateien nutzt. Diese Ausgabe kann Administratoren dabei helfen, ungewöhnliche Aktivitäten auf dem System festzustellen.

tiger

Die „tiger“-Sammlung ist ein Paket zur Analyse der Sicherheit des Rechnern. Die Vielzahl der Sicherheitstests, die „tiger“ durchführen kann, lässt sich vier Gruppen unterteilen:

Users: Dieser Test deckt Benutzerkonten ab, überprüft Mailweiterleitung, FTP-Zugänge und ähnliches.

Permission: Diese Auswahl überprüft Zugriffsrechte von Benutzern und Gruppen auf bestimmte wichtige Dateien und Verzeichnisse.

Config: Dieses Skripts sucht nach Schwachstellen und Fehlern in verbreiteten system- und anwendungsspezifischen Konfigurationsdateien.

System: Diese Tests sucht nach offenen gelöschten Dateien, Prozessen, die auf ankommende Verbindungen warten und andere „ungewöhnliche“ Dinge.

Full system check: Dies führt alle oben beschriebenen Tests aus.

tripwire

Tripwire ist ein Open Source-Programm, das die Integrität von Dateien überprüft. Beim ersten Aufruf erstellt Tripwire eine Datenbank mit Hashwerten der Systemdateien, die es bei zukünftigen Aufruf zum Vergleich heranzieht. Veränderungen am System lassen sich auf diese Weise leicht feststellen.

Die Standardinstallation von Tripwire enthält einen Regelsatz für Debian. Falls Sie diesen Regelsatz an eine andere Distribution oder ein anderes Betriebssystem anpassen möchten, finden Sie in der SLAD-Dokumentation für Entwickler und Administratoren weitere Informationen.

Snort

Snort dient dazu, in Echtzeit den Datenverkehr in IP-Netzwerken zu analysieren und zu überwachen und damit ein unerlaubtes Eindringen in diese Netzwerke frühzeitig zu erkennen und zu unterbinden. Es ist in der Lage, ein Vielzahl von Angriffe wie etwa Pufferüberläufe, verdeckte Portscans, CGI-Angriffe, SMB-Scans oder versuchte Betriebssystemerkennungen zu erkennen. Sobald ein Angriff entdeckt wurde, ist Snort ebenfalls zu Gegenmaßnahmen in der Lage, wie etwa der Unterbrechung der entsprechenden Verbindungen. Das SLAD-Plugin liest alle relevanten Snort-Meldungen aus einer MySQL-Datenbank und übermittelt sie an den OpenVAS-Client.

Snort-Installation Damit Sie das SLAD-Plugin für Snort nutzen können, muss Snort mit Unterstützung für MySQL installiert werden. Eine Anleitung dazu sollten Sie in der Dokumentation zu Ihrer Distributionen oder zu Snort selbst finden.

LMSensors

Mit dem LMSensors-Plugin können Sie Ereignisse der Hardware-Überwachung auslesen, wie etwa ein Öffnen des Servergehäuses. Der SLAD-Dienst unterstützt die Protokollierung dieser Sensordaten und wird eine entsprechende Meldung erzeugen, die den Vorfall beschreibt. Diese Möglichkeit wird von den meisten Mittelklasse-Servermainboards unterstützt, wie etwa Intel BX400 und neuer.

LogWatch

LogWatch liest Ereignisse aus den Systemprotokollen aus, etwa aus den unter `/var/log/` vorhandenen Dateien. Alle wichtigen Informationen wie Benutzeranmeldungen, SSH- und PAM-Sitzungen usw. werden gefiltert, zusammengefasst und an den SLAD-Dienst weitergegeben. Sie können auswählen, wie ausführlich diese Ergebnisse sein sollen und wie hoch der Grad der Zusammenfassung sein soll:

Low Erzeugt stark zusammengefasste, wenig detaillierte Meldungen.

Medium Erzeugt Meldungen mit einem mittleren Detailgrad.

High Ausführliche Meldungen mit dem niedrigsten Grad der Zusammenfassung.

TrapWatch

TrapWatch ist eine spezielle Version von LogWatch und protokolliert SNMP-Meldungen („Traps“) von Geräten. SNMP („Simple Network Management Protocol“) ist das am weitesten verbreitete Protokoll zur Verwaltung von Netzwerkhardware und wird von fast allen derzeit erhältlichen Netzwerkgeräten unterstützt. Ein SNMP-Trap ist eine Nachricht, die ein Netzwerkgerät sendet, um einen Zwischenfall (wie etwa einen unerwarteten Verbindungsabbruch, fehlgeschlagen Anmeldungsversuche usw.) zu melden. TrapWatch protokolliert diese Meldungen. Dies kann nützlich sein, um Veränderungen im Netzwerk festzustellen, wie etwa Rechner, die aus dem Netzwerk entfernt werden oder neu dazukommen. Unterstützung für Netscreen Firewall, HP Procure-Switches und Cisco-Hardware ist standardmäßig installiert. Falls Sie selbstdefinierte MIBs nutzen möchten, müssen Sie TrapWatch eventuell entsprechend konfigurieren. Bitte beachten Sie, dass einen SNMP Trap Handler benötigen, der die Trap-Meldungen im Systemprotokoll `syslog` festhält, falls Sie TrapWatch benutzen möchten.

8.2 Nikto

(von Michael Wiegand)

Nikto ist ein Open Source (GPL) Webserver-Scanner, der umfangreiche Sicherheitstests gegen Webserver durchführen kann. Zur Zeit überprüft Nikto auf über 3500 potentielle Sicherheitslücken, erkennt die Version von über 900 Servern und prüft auf versionsspezifische Probleme auf mehr als 250 Servern. Scandaten und Plugins werden regelmäßig aktualisiert und können automatisch aktualisiert werden.

OpenVAS ist in der Lage, eine installierte Nikto-Version zu erkennen und kann die Ergebnisse eines Nikto-Scans in die Scanergebnisse integrieren:

8.2.1 Voraussetzungen

Damit eine Nikto-Scan aus OpenVAS heraus ausgeführt werden kann, müssen die folgenden Voraussetzungen gegeben sein:

- Es muss eine Nikto-Version im Systempfad auf dem Server verfügbar sein. Die Integration von Nikto in OpenVAS ist für Nikto in der Version 2.0 und neuer optimiert; ältere Versionen werden vermutlich auch funktionieren.
- Der OpenVAS NVT für die Nikto-Integration muss verfügbar muss verfügbar und aktiviert sein. Sie finden den NVT im Abschnitt „CGI abuses“ in der NVT-Auswahl im Client.

8.2.2 Ausführung eines Nikto-Scans

Falls der Nikto-NVT verfügbar und aktiviert ist, wird er mit dem nächsten Scan ausgeführt. Die Ergebnisse, die Nikto zurückliefert, werden dann zusammen mit den Ergebnissen der anderen NVTs im Client dargestellt.

8.2.3 Nikto-Ergebnisse verstehen

Manche Webserver sind (absichtlich oder unabsichtlich) so konfiguriert, dass Sie auf Anfragen für nicht vorhandene Seiten mit einem anderen Statuscode als 404 antworten. Die kann genutzt werden, um derartige Anfragen von menschlichen Benutzern auf eine hilfreiche Seite (wie etwa eine Übersichtsseite) umzuleiten, verwirrt aber Sicherheitsanwendungen wie Nikto, die über diesen Mechanismus überprüfen, ob möglicherweise vertrauliche oder gefährliche Inhalt über diesen Webserver verfügbar sind.

Nikto kann dies in den meisten Fälle automatisch feststellen und wird standardmäßig keinen Scan gegen einen solchen Rechner starten. Sie können aber trotzdem einen Nikto-Scan erzwingen, indem Sie die Option *Force scan even without 404s* in den Voreinstellungen des NVTs aktivieren.

Falls Sie diese Option aktivieren, beachten Sie bitte, dass die Ergebnisse Ihres Nikto-Scans unter diesen Umständen so genannte „False Positives“ enthalten kann, also irrtümlich auf Sicherheitsprobleme hingewiesen wird; durch eine Konfiguration wie die oben beschriebene kann es passieren dass Nikto davon ausgeht, dass unter einer bestimmt Adresse potentiell vertrauliche oder gefährliche Daten verfügbar sind, obwohl der Server nur eine allgemeine Antwort auf die Anfrage gegeben.

Dies trifft besonders auf ältere Versionen von Nikto (vor 2.0) zu; aber auch mit neueren Versionen müssen Sie unter Umständen von Hand überprüfen, ob die von Nikto gefundenen Sicherheitslücken wirklich eine Gefahr darstellen oder nur das Ergebnis der Konfiguration des Webservers sind.

8.3 Ovaldi (OVAL Unterstützung in OpenVAS)

(von Michael Wiegand)

Die „Open Vulnerability and Assessment Language“ (OVAL) ist ein Standard, der unter anderem dazu genutzt werden kann, um sowohl bekannte Sicherheitslücken zu beschreiben als auch Tests, mit denen festgestellt werden kann, ob diese Sicherheitslücke auf einem Zielsystem existiert. Er nutzt XML-Dokumente, beispielsweise um potentiell unsichere Systeme und den aktuellen Zustand ihre Bestandteile zu beschreiben. Andere XML-Dokumente – die so genannten Definitionen – beschreiben einen bestimmten Zustand dieser Bestandteile, der als verwundbar angesehen wird. Im Gegensatz zu NASL beschreiben OVAL-Definitionen lediglich formal den Zustand eines verwundbaren System und sind nicht selbst Programme, die nach derartigen Verwundbarkeiten suchen.

Aus dem OVAL-Projekt heraus hat sich ovaldi entwickelt, eine Open Source Referenzimplementation eines Interpreters für OVAL Definitionen. Obwohl ovaldi zu Beginn nur lokale Systeme überprüfen konnte, ist es mittlerweile durch einen vom OpenVAS-Projekt erstellten Patch für ovaldi möglich, die OpenVAS gesammelten Informationen über entfernte System zu nutzen.

Unterstützung von ovaldi ist in OpenVAS ab der Version 2.0.0 verfügbar. Für die Unterstützung von ovaldi wird die SVN Revision 138 von ovaldi empfohlen. Auf der OpenVAS-Website erhalten die den für ovaldi notwendigen Patch und aktuelle Informationen über die ovaldi-Integration. Die aktuellste Version dieser Informationen ist unter http://www.openvas.org/integrated-tools.html verfügbar.

Mit ovaldi sind Sie in der Lage, auf mehrere Hundert zusätzliche Sicherheitstests zuzugreifen, die als OVAL-konforme Definitionen veröffentlicht wurden, wie etwa Sicherheitsmeldungen für die Red Hat Enterprise Linux Distribution. Bitte beachten Sie, dass die Integration von ovaldi in OpenVAS zur Zeit nur einen Teile der in OVAL möglichen Tests abdeckt. Die Unterstützung für OVAL Tests wird sich im Zuge der weiteren ovaldi-Integration erweitern.

Nach dem Sie die Unterstützung für OVAL Definitionen erfolgreich aktiviert haben, sind diese im OpenVAS-Client in der Familie „OVAL definitions“ verfügbar. Die meisten Definitionen werden einen der folgenden Werte zurückgeben: „true“, „false“ or „unknown“.

Diese Werte sind wie folgt definiert:

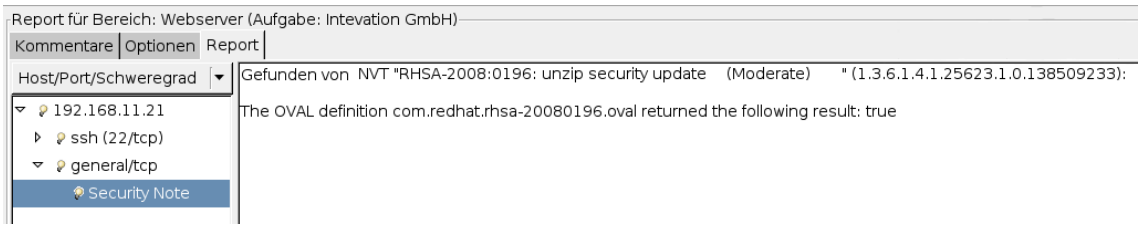
true Die von ovaldi ausgewerteten Tests deuten mit hoher Wahrscheinlichkeit darauf hin, dass die in der Definition beschriebene Sicherheitslücke auf dem System vorhanden ist.

false Die von ovaldi ausgewerteten Tests deutet darauf hin, dass es unwahrscheinlich ist, dass die in der Definition beschriebene Sicherheitslücke auf dem System vorhanden ist.

unknown Die von ovaldi ausgewerteten Tests ergaben kein klares Ergebnis bzw. ovaldi war nicht in der Lage, alle für diese Überprüfung notwendigen Tests auszuführen.

Beachten Sie, dass eine große Anzahl von Tests den Wert „unknown“ zurückgeben wird, bis die Unterstützung für OVAL in OpenVAS weit genug fortgeschritten ist.

Die Ergebnisse der OVAL Definitionen wird Ihnen auf die gleiche Weise angezeigt wie die Ergebnisse der übrigen NVTs, so dass Sie sich bequem über die Ergebnisse informieren können, ohne OpenVAS-Client zu verlassen.



Weitere Informationen über das OVAL-Projekt und die OVAL-Sprache finden Sie unter http://oval.mitre.org/. Die Projektseite für ovaldi finden Sie unter http://sourceforge.net/projects/ovaldi/.

9 Programmierung von Network Vulnerability Tests

Die von OpenVAS genutzten Sicherheitstests („Network Vulnerability Tests“, NVTs) sind in der Skriptsprache NASL geschrieben. NASL („Nessus Attack Scripting Language“) wurde ursprünglich für den Sicherheitsscanner Nessus von Renaud Deraison entworfen.

Das Ziel beim Entwurf von NASL war es, ein Sprache zu schaffen, die es auch Anwendern mit begrenzter Programmiererfahrung erlaubte, innerhalb von kurzer Zeit Sicherheitstests zu schreiben und es Anwendern ermöglichte, auf einfache Weise neue Sicherheitstests zu ihrer bestehenden Installation hinzuzufügen ohne sich Gedanke um die Portabilität ihrer Skripte machen zu müssen.

Die erste Version von NASL (auch als NASL1 bekannt) wurde 1998 von Renaud Deraison geschaffen. Michel Arboi und Renaud Deraison entwickelten 2002 einen verbesserten NASL parser, der weitere Funktionen und Operatoren verfügbar machte. Diese verbesserte Version wird im Allgemeinen als NASL2 bezeichnet.

Sofern nicht anders angegeben, beschreibt das vorliegende Kompendium die in OpenVAS verwendete Version NASL2.

Die NASL Syntax wurde von C inspiriert. Anwender mit Erfahrung in C oder verwandten Programmiersprachen sollten in der Lage sein, sich die Grundlagen der NASL-Programmierung in relativ kurzer Zeit zu erarbeiten.

Seit OpenVAS 2.0.0 steht auch Unterstützung für die Open Vulnerability and Assessment Language (OVAL) zur Verfügung, wie in Abschnitt 8.3 beschrieben. Das heißt, dass OpenVAS nun in der Lage ist, auch in OVAL verfasste Sicherheitstests auszuführen. Auch wenn die Unterstützung für OVAL zur Zeit noch begrenzt ist, können Sie OVAL als Alternative zu NASL in Betracht ziehen, wenn Sie NVTs schreiben. Mehr Informationen über OVAL erhalten Sie unter: http://oval.mitre.org/language/about/index.html.

9.1 Aufbau eines NASL Skripts

Alle NASL-Skripte müssen gewissen Informationen enthalten, anhand derer das Skript von anderen Skripten eindeutig unterschieden werden kann und mit denen auf das Skript verwiesen werden kann. Diese Informationen sind im dem als `description` oder `register` bezeichneten Teil des Skripts enthalten, den alle Skripte enthalten müssen. Im Allgemeinen steht dieser Teil am Beginn eines jeden NASL-Skripts.

Ein einfaches NASL-Skript könnte wie folgt beginnen:

```
#
# This is an example NASL script.
#

if(description)
{
 script_oid("1.3.6.1.4.1.25623.1.0.12345")
 script_version ("1.2");
 name["english"] = "Foo Bar 2.5 vulnerability";
 script_name(english:name["english"]);
```

```
 desc["english"] = "
   This plugin checks for the vulnerability in the Foo Bar 2.5 server
   component as described in CVE 2009-4321.

   Risk factor : None";

 script_description(english:desc["english"]);

 summary["english"] = "Check for vulnerability in Foo Bar 2.5";
 script_summary(english:summary["english"]);

 script_copyright(english:"This script is under GPLv2+");

 ...

 exit(0);
}

...
```

Die Beschreibung des NVTs ist in dem Block `if (description)` enthalten, der dem OpenVAS Server den Abruf dieser Informationen ermöglicht. Wenn der Server einen neuen NVT zum ersten Mal einliest, wird das Skript aufgerufen, während die globale Variable `description` auf `TRUE` (1) gesetzt ist. Die Informationen, die das Skript liefert, werden vom Server in einem Unterverzeichnis des NVT-Verzeichnisses namens `.desc` zwischengespeichert. Wenn das Skript während eines Scans aufgerufen wird, ist die globale Variable `description` auf `FALSE` (0) gesetzt.

Eine vollständige Liste aller NASL-Befehle, die in der Beschreibung eines Skriptes verwendet werden können, finden Sie in Abschnitt 9.3 der Dokumentation der NASL API.

9.2 NASL Syntax

9.2.1 Kommentare

Kommentare beginnen in NASL mit dem Zeichen `#`. Wenn eine Zeile ein `#` enthält, wird der darauf folgenden Teil der Zeile vom NASL Interpreter ignoriert.

Beispiel:

```
# This is a comment.
a = b + c; # This is a comment as well.
```

9.2.2 Variablen und Deklarationen

Variablen werden in NASL implizit deklariert; das heißt, dass es nicht notwendig ist, Variablen vor der Benutzung zu deklarieren. Sie können aber das Schlüsselwort `local_var` benutzen, um eine Variable als lokale Variable einer bestimmten Funktion zu deklarieren und so Kollisionen mit externen Variablen zu vermeiden. Wenn nichts anderes angeben wurde, behandelt NASL eine Variablen als lokale Variable in dem Kontext, in dem sie deklariert wurde; wenn Sie eine Variable als global deklarieren möchten, müssen Sie das Schlüsselwort `global_var` verwenden.

Im Gegensatz zu C (und anderen Sprachen) muss der Typ der Variablen nicht im Voraus deklariert werden; der NASL Interpreter wird bei der ersten Zuweisung den entsprechenden Datentyp auswählen. Die Speicherverwaltung wird ebenfalls automatisch vom Interpreter übernommen.

9.2.3 Datentypen

- Integers
- Strings
- Arrays
- Booleans

9.2.4 Zahlen und Zeichenketten

9.2.5 Funktionsargumente

9.2.6 Schleifen

- for
- foreach
- while
- repeat .. until
- break
- continue
- return

9.2.7 Benutzerdefinierte Funktionen

9.2.8 Operatoren

- =
- `[]`
- +
- -
- *
- /
- %
- **
- ++
- +=
- -=
- *=

- /=
- %=
- «=
- »=
- »>=
- ><
- >!<
- =
- !
- ==
- !=
- >
- >=
- <
- <=
- !
- &&
- ||
- ~
- &
- |
- ^
- <<
- >>
- >>>

Operatorrangfolge

9.3 NASL API Dokumentation

9.3.1 Vordefinierte Konstanten

9.3.2 Built-In Funktionen

Socket Manipulation

- `open_sock_tcp()`
- `open_sock_udp()`

- open_priv_sock_tcp()
- open_priv_sock_udp()
- close()
- shutdown()
- recv()
- recv_line()
- send()

Netzwerk

- start_denial()
- end_denial()
- get_port_transport()
- get_source_port()
- get_tcp_port_state()
- get_udp_port_state()
- islocalhost()
- islocalnet()
- join_multicast_group()
- leave_multicast_group()
- scanner_add_port()
- scanner_get_port()
- this_host_name()

FTP

- ftp_log_in()
- ftp_get_pasv_port()

HTTP

- is_cgi_installed()
- http_get()
- http_head()
- http_post()
- cgibin()
- http_delete()
- http_close_socket()
- http_open_socket()
- http_recv_headers()
- http_put()

Packet

- forge_ip_packet()
- get_ip_element()
- set_ip_elements()
- dump_ip_packet()
- forge_tcp_packet()
- set_tcp_elements()
- send_packet()
- pcap_next()
- get_tcp_element()
- forge_udp_packet()
- set_udp_elements()
- get_udp_elements()
- forge_icmp_packet()
- get_icmp_element()
- set_icmp_elements()
- forge_igmp_packet()
- dump_tcp_packet()
- dump_udp_packet()

Utilities

- this_host()
- get_host_name()
- get_host_ip()
- get_host_open_port()
- get_port_state()
- telnet_init()
- tcp_ping()
- getrpcport()

Manipulation von Zeichenketten

- `ereg()`
- `ereg_replace()`
- `egrep()`
- `crap()`
- `string()`
- `strlen()`
- `raw_string()`
- `tolower()`
- `chomp()`
- `display()`
- `eregmatch()`
- `hex()`
- `hexstr()`
- `insstr()`
- `int()`
- `match()`
- `ord()`
- `str_replace()`
- `strcat()`
- `stridx()`
- `strstr()`
- `split()`
- `substr()`
- `toupper()`

Wissensbasis

- `get_kb_item()`
- `set_kb_item()`
- `get_kb_list()`
- `replace_kb_item()`
- `replace_or_set_kb_item()`

NVT Beschreibung

- `script_id()`
- `script_oid()`

  (von Tim Brown)

  Diese Funktion soll die Funktion `script_id` ablösen, die derzeit verwendet wird, um NASL-Skripte eindeutig zu identifizieren. Der Gedanke hinter dieser Ablösung ist, dass `script_id` nur einen einzigen globalen Bereich zur Verfügung steht. Da es bereits Pläne von mehreren Organisationen gibt, eigene NVTs und Feed Services anzubieten, wurde ein neuer OID-basierter Bereich eingeführt, der einfacher durch mehrere Parteien genutzt werden kann, ohne dass die Gefahr von Kollisionen besteht.

  Die derzeitige Implementierung verhält sich wie folgt: Jedes NVT, das einen `script_id`-Aufruf verwendet, erhält automatisch eine OID aus dem für „alte“ NVTs reservierten Bereich. Dieser OID-Bereich hat den Präfix „1.3.6.1.4.1.25623.1.0“. Der OpenVAS zugewiesene OID-Bereich wird zur Zeit von Tim Brown verwaltet.

  Sowohl der Client als auch der Server wurden aktualisiert, um diese Funktionalität zu unterstützen. Sie können feststellen, ob dies der Fall ist, in dem Sie überprüfen, ob die globale Konstanten `OPENVAS_NASL_LEVEL` größer als 2206 ist.

  `script_oid` sollte auf die folgende Weise aufgerufen werden:

```
...
if(description)
{
  if (OPENVAS_NASL_LEVEL >= 2206)
  {
    script_oid("1.3.6.1.4.1.25623.1.0.90010");
  }
  else
  {
    script_id(90010);
  }
...
```

- `script_version()`
- `script_name()`
- `script_description()`
- `script_summary()`
- `script_category()`
- `script_copyright()`
- `script_family()`
- `script_dependencies()`
- `script_cve_id()`
- `script_require_ports()`
- `script_require_keys()`
- `script_exclude_keys()`
- `scanner_status()`

- `script_get_preference()`
- `script_add_preference()`
- `script_bugtraq_id()`
- `script_dependencie()`
- `script_get_preference_file_content()`
- `script_get_preference_file_location()`
- `script_require_udp_ports()`
- `script_timeout()`

Berichtsfunktionen

- `security_warning()`
- `security_hole()`
- `security_info()`
- `security_note()`
- `log_message()`
- `debug_message()`

Kryptografische Funktionen

- `HMAC_DSS()`
- `HMAC_MD2()`
- `HMAC_MD4()`
- `HMAC_MD5()`
- `HMAC_RIPEMD160()`
- `HMAC_SHA()`
- `HMAC_SHA1()`
- `MD2()`
- `MD4()`
- `MD5()`
- `RIPEMD160()`
- `SHA()`
- `SHA1()`

Verschiedene Funktionen

- `cvsdate2unixtime()`
- `defined_func()`
- `dump_ctxt()`
- `func_has_arg()`
- `func_named_args()`
- `func_unnamed_args()`
- `gettimeofday()`
- `isnull()`
- `localtime()`
- `max_index()`
- `mktime()`
- `safe_checks()`
- `sleep()`
- `type_of()`
- `usleep()`
- `unixtime()`
- `make_list()`
- `make_array()`
- `get_preference()`

„Unsichere“ Funktionen

Bitte beachten Sie: Unter Nessus wurden die folgenden Funktionen als „unsicher“ eingestuft, da sie auf das Dateiesystem zugreifen. Die Funktionen standen nur einer Auswahl von vertrauenswürdiger Plugins zu Verfügung. Da OpenVAS einen anderen Ansatz verfolgt und eine gleichzeitige Ausführung von vertrauenswürdigen und nicht vertrauenswürdigen Plugins nicht zulässt, stehen diese Funktion allen Plugins zur Verfügung. Die Einstufung als „unsichere“ Funktion geschieht an dieser Stelle lediglich aus historischen Gründe. Gleichwohl sollten sich Entwickler darüber im Klaren sein, dass diese Funktionen auf dem OpenVAS ausführenden System Schaden anrichten können. Es wird erwartet, dass NASL-Entwickler beim Einsatz dieser Funktionen die notwendige Vorsicht walten lassen.

- `find_in_path()`
- `file_close()`
- `file_open()`
- `file_read()`
- `file_seek()`
- `file_stat()`
- `file_write()`

- `fread()`
- `fwrite()`
- `get_tmp_dir()`
- `unlink()`
- `pread()`

9.3.3 Funktionsbibliotheken

Um die „eingebauten“ Funktionen zu erweitern ist es möglich, über so genannte „include“-Dateien zusätzliche Funktionalität aus Funktionsbibliotheken einzubinden. Diese Dateien haben per Konvention die Endung „.inc“ und liegen im NVT-Verzeichnis. Bevor Sie eigene Funktionen für Ihre NASL-Skripte schreiben, sollten Sie zunächst überprüfen, ob nicht die von Ihnen benötigte Funktionalität nicht schon von einer Funktionsbibliotheken bereitgestellt wird. Wenn Sie meinen, dass die von Ihnen geschriebenen Funktionen auch anderen Entwicklern nützlich sein könnten, sollten Sie in Betracht ziehen, diese den verfügbaren Funktionsbibliotheken hinzuzufügen.

Eine ganze Anzahl von Funktionen sind bereits über die Funktionsbibliotheken verfügbar; die folgende Liste enthält die Namen der aktuell mit OpenVAS ausgelieferten Funktionsbibliotheken und die von ihnen definierten Funktionen. Berücksichtigen Sie bitte, dass diese Information ständigen Änderungen unterliegt.

backport.inc get_backport_banner(), get_php_version()

debian_package.inc deb_check(), deb_str_cmp(), deb_ver_cmp()

default_account.inc check_account(), _check_telnet(), recv_until()

dump.inc dump(), hexdump(), isprint (), line2string ()

ftp_func.inc ftp_authenticate(), ftp_close(), ftp_pasv(), ftp_recv_data(), ftp_recv_line(), ftp_recv_listing(), ftp_send_cmd(), get_ftp_banner()

global_settings.inc debug_print(), log_print()

http_func.inc can_host_asp(), can_host_php(), cgi_dirs(), check_win_dir_trav(), do_check_win_dir_trav(), get_http_banner(), get_http_port(), headers_split(), hex2dec(), __hex_value(), http_40x(), http_is_dead(), http_recv(), http_recv_body(), http_recv_headers2(), http_recv_length(), http_send_recv(), php_ver_match()

http_keepalive.inc check_win_dir_trav_ka(), enable_keepalive(), get_http_page(), http_keepalive_check_connection(), http_keepalive_enabled(), http_keepalive_recv_body(), http_keepalive_send_recv(), is_cgi_installed_ka(), on_exit()

imap_func.inc get_imap_banner ()

misc_func.inc add_port_in_list(), base64(), base64_code(), base64_decode(), cvsdate2unixtime(), dec2hex(), get_mysql_version(), get_rpc_port(), get_service_banner_line(), get_unknown_banner(), hex2raw(), known_service(), pow2(), rand_str(), register_service(), replace_or_set_kb_item(), report_service(), service_is_unknown(), set_mysql_version(), set_unknown_banner()

netop.inc ip_dot2raw(), ip_raw2dot(), netop_banner_items(), netop_check_and_add_banner(), netop_each_found(), netop_kb_derive(), netop_log_detected(), netop_product_ident(), netop_spacepad(), netop_zeropad()

network_func.inc htonl(), htons(), ip_checksum(), is_private_addr(), ms_since_midnight(), ntohl(), test_udp_port()

nfs_func.inc cwd(), mount(), open(), padsz(), read(), readdir(), rpclong(), rpcpad(), str2long(), umount()

nntp_func.inc nntp_article(), nntp_auth(), nntp_connect(), nntp_make_id(), nntp_post()

pingpong.inc udp_ping_pong()

pkg-lib-deb.inc isdpkgvuln()

pop3_func.inc get_pop3_banner ()

qpkg.inc qpkg_check(), qpkg_cmp(), qpkg_ver_cmp()

revisions-lib.inc isdigit(), revcomp()

slackware.inc slack_elt_cmp(), slack_ver_cmp(), slackware_check()

slad_ssh.inc slad_ssh_login ()

smbcl_func.inc bin_dword(), bin_word(), fileread(), GetPEFileVersion (), GetPEProductVersion (), get_windir(), is_domain(), PEVersion(), smbclientavail(), smbgetdir(), smbgetfile(), smbversion()

smb_hotfixes.inc hotfix_check_dhcpserver_installed(), hotfix_check_domain_controler(), hotfix_check_excel_version(), hotfix_check_exchange_installed(), hotfix_check_iis_installed(), hotfix_check_nt_server(), hotfix_check_office_version(), hotfix_check_outlook_version(), hotfix_check_powerpoint_version(), hotfix_check_sp(), hotfix_check_wins_installed(), hotfix_check_word_version(), hotfix_check_works_installed(), hotfix_data_access_version(), hotfix_get_commonfilesdir() hotfix_get_programfilesdir(), hotfix_get_systemroot(), hotfix_missing()

smtp_func.inc get_smtp_banner(), smtp_close(), smtp_from_header(), smtp_open(), smtp_recv_banner(), smtp_recv_line(), smtp_send_port(), smtp_send_socket(), smtp_to_header()

ssh_func.inc base64decode(), check_pattern(), crypt(), decrypt(), derive_keys(), dh_gen_key(), dh_valid_key(), get_data_size(), get_ssh_banner(), get_ssh_error(), get_ssh_server_version(), get_ssh_supported_authentication(), getstring(), init(), is_sshd_bugged(), kb_ssh_login(), kb_ssh_passphrase(), kb_ssh_password(), kb_ssh_privatekey(), kb_ssh_publickey(), kb_ssh_transport(), kex_packet(), load_array_from_kb(), load_data_from_kb(), load_intarray_from_k load_int_from_kb(), mac_compute(), ntol(), packet_payload(), putbignum(), putstring(), raw_int32(), raw_int8(), recv_ssh_packet(), register_array_in_kb(), register_data_in_kb(), register_intarray_in_kb(), register_int_in_kb(), reuse_connection_init(), send_ssh_packet(), set_ssh_error(), ssh_close_channel(), ssh_close_connection(), ssh_cmd(), ssh_cmd_error(), ssh_dss_verify(), ssh_exchange_identification(), ssh_hex2raw(), ssh_kex2(), ssh_login(), ssh_login_or_reuse_connection(), ssh_open_channel(), ssh_recv(), ssh_reuse_connection(), ssh_rsa_verify(), ssh_userauth2(), update_window_size()

telnet_func.inc get_telnet_banner(), set_telnet_banner()

tftp.inc tftp_get(), tftp_put()

ubuntu.inc deb_str_cmp(), ubuntu_check(), ubuntu_ver_cmp()

uddi.inc create_uddi_xml ()

version_func.inc find_bin(), get_bin_version(), get_string_version(), version_is_equal(), version_is_greater(), version_is_greater_equal(), version_is_less(), version_is_less_equal(), version_test()

9.3.4 Wissensbasis

Um den Informationsaustausch zwischen verschiedenen NVTs zu erleichtern und den Scanablauf zu beschleunigen, werden von NVTs gesammelte Informationen in einer so genannten Wissensbasis („Knowledge Base“, KB) abgelegt. Dies ermöglicht es NVTs, auf den Ergebnissen anderer NVTs aufzubauen und hilft, doppelte Scans zu vermeiden.

Die folgende Liste enthält bekannte Einträge in die Wissensbasis und die Namen der NASL/NES-Skripte, die diese Werte setzten. Berücksichtigen Sie bitte, dass diese Liste vermutlich lückenhaft ist und ständigen Änderungen unterworfen ist.

Amanda/running (amanda_detect.nasl): 1 = Amanda läuft auf dem Zielrechner

Amanda/version (amanda_version.nasl):

Amap/*/FullBanner (amap.nasl):

Amap/*/PrintableBanner (amap.nasl):

Amap/*/Svc (amap.nasl):

bind/version (bind_version.nasl): Version des entfernten BIND-Servers

cfengine/running (cfengine_detect.nasl):

cheopsNG/password (cheopsNG_detect.nasl):

cheopsNG/unprotected (cheopsNG_detect.nasl):

CVE-2003-1011 (apple-sa-2004-08-09.nasl):

ebola/banner/* (find_service2.nasl):

fake_identd/113 (w32_spybot_worm_variant.nasl):

fake_identd/* (slident.nasl, find_service1.nasl, ident_backdoor.nasl):

FindService/tcp/*/get_http (doublecheck_std_services.nasl, apache_SSL_complain.nasl):

FindService/tcp/*/help (doublecheck_std_services.nasl, find_service2.nasl, find_service_3digits.nasl):

FindService/tcp/*/spontaneous (find_service_3digits.nasl, doublecheck_std_services.nasl):

fsp/banner/* (fsp_detection.nasl):

ftp/backdoor (ftp_kibuv_worm.nasl):

ftp/fw1ftpd (ftpserver_detect_type_nd_version.nasl): 1 = Auf dem Zielrechner läuft FW/1 FTPd

ftp/login (logins.nasl): Benutzername für FTP-Anmeldung

ftp/msftpd (ftpserver_detect_type_nd_version.nasl): 1 = Auf dem Zielrechner läuft IIS FTPd

ftp/ncftpd (ftpserver_detect_type_nd_version.nasl): 1 = Auf dem Zielrechner läuft NcFTPd

ftp/password (logins.nasl): Passwort für FTP-Anmeldung

ftp/*/AnyUser (DDI_FTP_Any_User_Login.nasl):

ftp/*/backdoor (ftp_kibuv_worm.nasl):

ftp/*/syst (find_service_3digits.nasl):

ftp/vxftpd (ftpserver_detect_type_nd_version.nasl):

ftp/writeable_dir (ftp_writeable_directories.nasl, logins.nasl, ftp_write_dirs.nes):

ftp/wuftpd (ftpserver_detect_type_nd_version.nasl):

global_settings/debug_level (global_settings.nasl):

global_settings/experimental_scripts (global_settings.nasl):

global_settings/http_user_agent (global_settings.nasl):

global_settings/log_verbosity (global_settings.nasl):

global_settings/network_type (global_settings.nasl):

global_settings/report_paranoia (global_settings.nasl):

global_settings/report_verbosity (global_settings.nasl):

global_settings/thorough_tests (global_settings.nasl):

Host/accept_lsrr (source_routed.nasl):

Host/dead (3com_nbx_voip_netset_detection.nasl, blackice_dos.nasl, dont_scan_printers.nasl, ftp_w98_devname_dos.nasl, fw1_udp_DoS.nasl, http_w98_devname_dos.nasl, jolt2.nasl, jolt.nasl, labrea.nasl, linux_icmp_sctp_DoS.nasl, open_all_ports_DoS.nasl, p-smash.nasl, spank.nasl, TLD_wildcard.nasl, vxworks_ftpdDOS.nasl): 1 = Der Zielrechner antwortet nicht mehr

Host/Debian/dpkg-l (ssh_get_info.nasl):

Host/firewall (securemote_info_leak.nasl, securemote.nasl): (Name des Firewalls) Der Zielrechner ist ein Firewall

Host/full_scan (amap.nasl, netstat_portscan.nasl, nmap.nasl, snmpwalk_portscan.nasl, openvas_tcp_scanner.nes, synscan.nes):

Host/ident_scanned (ident_process_owner.nasl, netstat_portscan.nasl, nmap.nasl):

Host/num_ports_scanned (openvas_tcp_scanner.nes, synscan.nes):

Host/OS (nmap.nasl, nmap_wrapper.nes):

Host/OS/ntp (ntp_open.nasl):

Host/ping_failed (nmap.nasl, nmap_wrapper.nes):

Host/processor/ntp (ntp_open.nasl):

Host/protocol_scanned (ip_protocol_scan.nasl):

Host/scanned (amap.nasl, netstat_portscan.nasl, nmap.nasl, snmpwalk_portscan.nasl, nmap_tcp_connect.nes, nmap_wrapper.nes, synscan.nes, openvas_tcp_scanner.nes, ike-scan.nasl): 1 = Der Portscan des Zielrechners ist abgeschlossen

Host/scanners/amap (amap.nasl):

Host/scanners/ike-scan (ike-scan.nasl):

Host/scanners/netstat (netstat_portscan.nasl):

Host/scanners/nmap (nmap.nasl):

Host/scanners/openvas_tcp_scanner (openvas_tcp_scanner.nes):

Host/scanners/snmpwalk (snmpwalk_portscan.nasl):

Host/scanners/synscan (synscan.nes):

Host/tcp_reverse_lsrr (source_routed.nasl):

Host/tcp_seq_idx (nmap.nasl):

Host/tcp_seq (nmap.nasl, nmap_wrapper.nes):

Host/udp_scanned (amap.nasl, netstat_portscan.nasl, nmap.nasl, snmpwalk_portscan.nasl, nmap_wrapper.nes):

http/auth (logins.nasl): Benutzername für HTTP-Anfragen

http/login (logins.nasl):

http/password (logins.nasl): Passwort für HTTP-Anfragen

Hydra/cisco-enable/* (hydra_cisco_enable.nasl):

Hydra/cisco/* (hydra_cisco.nasl):

Hydra/cvs/* (hydra_cvs.nasl):

Hydra/ftp/* (hydra_ftp.nasl):

Hydra/http/* (hydra_http.nasl):

Hydra/http-proxy/* (hydra_http_proxy.nasl):

Hydra/icq/* (hydra_icq.nasl):

Hydra/imap/* (hydra_imap.nasl):

Hydra/ldap/* (hydra_ldap.nasl):

Hydra/mssql/* (hydra_mssql.nasl):

Hydra/nntp/* (hydra_nntp.nasl):

Hydra/pcnfs/* (hydra_pcnfs.nasl):

Hydra/rexec/* (hydra_rexec.nasl):

Hydra/sapr3/* (hydra_sapr3.nasl):

Hydra/smtp-auth/* (hydra_smtp_auth.nasl):

Hydra/snmp/* (hydra_snmp.nasl):

Hydra/socks5/* (hydra_socks5.nasl):

Hydra/ssh2/* (hydra_ssh2.nasl):

Hydra/telnet/* (hydra_telnet.nasl):

Hydra/vnc/* (hydra_vnc.nasl):

Ident/*/* (nmap.nasl):

Ident/tcp/* (ident_process_owner.nasl, netstat_portscan.nasl):

iis/global.asa.download (DDI_GlobalASA_Retrieval.nasl):

imap/banner/* (find_service2.nasl):

imap/login (logins.nasl): Benutzername für IMAP-Anmeldung

imap/password (logins.nasl): Passwort für IMAP-Anmeldung

imap/*/Cyrus (cyrus_imap_prelogin_overflow.nasl):

IPProtocol/* (ip_protocol_scan.nasl):

kazaa/username (kazaa_morpheus_detect.nasl):

mssql/SQLVersion (mssql_version.nasl):

mssql/udp/1434 (mssql_ping.nasl):

MSSQL/UDP/Ping (mssql_ping.nasl):

Nmap/*/svc (nmap.nasl):

Nmap/*/version (nmap.nasl):

nntp/local_distrib (nntp_info.nasl):

nntp/login (logins.nasl):

nntp/password (logins.nasl):

nntp/*/cancel (nntp_info.nasl):

nntp/*/noauth (nntp_info.nasl):

nntp/*/posted (nntp_info.nasl):

nntp/*/posting (nntp_info.nasl):

nntp/*/ready (nntp_info.nasl):

nntp/*/supersed (nntp_info.nasl):

nntp/x_no_archive (nntp_info.nasl):

NTP/Running (ntp_open.nasl):

oracle_tnslsnr/*/version (oracle_tnslsnr_version.nasl):

pop2/login (logins.nasl):

pop2/password (logins.nasl):

pop3/login (logins.nasl):

pop3/password (logins.nasl):

/rip/*/broken_source_port (rip_detect.nasl):

/rip/*/version (rip_detect.nasl):

Secret/hydra/logins_file (hydra_options.nasl):

Secret/hydra/passwords_file (hydra_options.nasl):

Secret/SSH/login (ssh_authorization.nasl):

Secret/SSH/passphrase (ssh_authorization.nasl):

Secret/SSH/password (ssh_authorization.nasl):

Secret/SSH/privatekey (ssh_authorization.nasl):

Secret/SSH/publickey (ssh_authorization.nasl):

Services/data_protector/build (hp_data_protector_installed.nasl):

Services/data_protector/version (hp_data_protector_installed.nasl):

Services/three_digits (find_service.nes):

Services/unknown (find_service.nes): Port des unbekannten Dienstes

Services/wrapped (find_service.nes):

Services/www/*/broken (http_func.inc, no404.nasl):

Services/www/*/embedded (3com_nbx_voip_netset_detection.nasl, ciscoworks_detect.nasl, clearswift_mimesweeper_smtp_ cobalt_web_admin_server.nasl, DDI_Cabletron_Web_View.nasl, DDI_F5_Default_Support.nasl, embedded_web_server_detect.nasl, imss_detect.nasl, interspect_detect.nasl, intrushield_console_detect.nasl, iwss_detect.nasl, linksys_multiple_vulns.nasl, raptor_detect.nasl, securenet_provider_detect.nasl, sitescope_management_server.nasl, sun_cobalt_adaptive_firewall_detect.nasl, tmcm_detect.nasl, websense_detect.nasl, xedus_detect.nasl):

Services/www/*/kerio_wrf (kerio_wrf_management_detection.nasl):

Services/www/*/working (http_func.inc):

Settings/disable_cgi_scanning (global_settings.nasl):

Settings/ExperimentalScripts (global_settings.nasl):

Settings/ThoroughTests (global_settings.nasl):

sip/banner/5060 (sip_detection.nasl):

SMB/domain_filled/0 (smb_authorization.nasl):

SMB/DOMAIN (smbcl_func.inc):

SMB/dont_send_in_cleartext (logins.nasl):

SMB/dont_send_ntlmv1 (logins.nasl):

SMB/ERROR (smbcl_func.inc):

SMB/FILEVERSION/* (smbcl_func.inc):

SMB/login_filled/0 (smb_authorization.nasl):

SMB/name (netbios_name_get.nasl): NetBIOS-Name des Zielrechners

SMB/OS (smbcl_func.inc):

SMB/password_filled/0 (smb_authorization.nasl):

SMB/PRODUCTVERSION/* (smbcl_func.inc):

SMB/samba (netbios_name_get.nasl): 1 = Auf dem entfernen Zielrechner läuft Samba

SMB/SERVER (smbcl_func.inc):

SMB/smbclient (smbcl_func.inc):

SMB/transport (cifs445.nasl):

SMB/username (netbios_name_get.nasl):

SMB/WinXP/ServicePack (smb_reg_service_pack_XP.nasl):

SMB/workgroup (netbios_name_get.nasl): Name der Arbeitsgruppe/Domäne des Zielrechners

SMTP/3comnbx (smtpserver_detect.nasl):

SMTP/domino (smtpscan.nasl, smtpserver_detect.nasl):

SMTP/exim (smtpscan.nasl, smtpserver_detect.nasl):

SMTP/firewall-1 (smtpscan.nasl, smtpserver_detect.nasl):

SMTP/intermail (smtpscan.nasl, smtpserver_detect.nasl):

SMTP/interscan (smtpscan.nasl):

SMTP/microsoft_esmtp_5 (smtpscan.nasl, smtpserver_detect.nasl): 1 = Auf dem Zielrechner läuft MS SMTP 5

smtp/*/broken (check_smtp_helo.nasl):

smtp/*/denied (check_smtp_helo.nasl):

smtp/*/helo (check_smtp_helo.nasl):

smtp/*/real_banner (smtpscan.nasl):

smtp/*/temp_denied (check_smtp_helo.nasl):

SMTP/postfix (smtpscan.nasl, smtpserver_detect.nasl): 1 = Auf dem Zielrechner läuft Postfix

SMTP/postoffice (smtpscan.nasl):

SMTP/qmail (smtpscan.nasl, smtpserver_detect.nasl): 1 = Auf dem Zielrechner läuft qmail

SMTP/sendmail (smtpscan.nasl): 1 = Auf dem Zielrechner läuft Sendmail

SMTP/sendmail (smtpserver_detect.nasl): 1 = Auf dem Zielrechner läuft Sendmail

SMTP/snubby (smtpserver_detect.nasl):

SMTP/wrapped (check_smtp_helo.nasl): 1 = Sendmail auf dem Zielrechner ist „wrapped“

SMTP/wrapped (smtpserver_detect.nasl): 1 = Sendmail auf dem Zielrechner ist „wrapped“

SMTP/xmail (smtpscan.nasl, smtpserver_detect.nasl):

SMTP/zmailer (smtpserver_detect.nasl):

SNMP/community (snmp_default_communities.nasl): Name einer gültigen SNMP-Community

SNMP/port (snmp_default_communities.nasl):

SNMP/running (snmp_detect.nasl):

socks/*/auth/* (socks.nasl):

SSH/banner/* (ssh_detect.nasl):

ssh/login/freebsdpatchlevel (gather-package-list.nasl):

ssh/login/freebsdpkg (gather-package-list.nasl):

ssh/login/freebsdrel (gather-package-list.nasl):

ssh/login/gentoo (gather-package-list.nasl):

ssh/login/packages (gather-package-list.nasl):

ssh/login/pkg (gather-package-list.nasl):

ssh/login/release (gather-package-list.nasl):

ssh/login/rpms (gather-package-list.nasl):

ssh/login/slackpack (gather-package-list.nasl):

ssh/login/solosversion (gather-package-list.nasl): Solaris OS Version wie von uname -r berichtet

ssh/login/solhardwaretype (gather-package-list.nasl): Solaris Hardware Typ wie von uname -p berichtet

ssh/login/solpackages (gather-package-list.nasl): Solaris Pakete wie von pkginfo berichtet

ssh/login/solpatches (gather-package-list.nasl): Solaris Patches wie von showrev -p berichtet

ssh/login/uname (gather-package-list.nasl, ssh_get_info.nasl):

SSH/supportedauth/* (ssh_detect.nasl):

SSH/textbanner/* (ssh_detect.nasl):

TCPScanner/NbPasses (openvas_tcp_scanner.nes):

TCPScanner/OpenPortsNb (openvas_tcp_scanner.nes):

TCPScanner/ClosedPortsNb (openvas_tcp_scanner.nes):

TCPScanner/FilteredPortsNb (openvas_tcp_scanner.nes):

TCPScanner/RSTRateLimit (openvas_tcp_scanner.nes):

tftp/get_file (tftp_grab_file.nes):

tftp/backdoor (tftpd_backdoor.nasl):

tftp/*/backdoor (tftpd_backdoor.nasl):

tftp/*/filecontent/* (tftpd_dir_trav.nasl):

tftp/*/filename/* (tftpd_dir_trav.nasl):

tftp/*/get_file (tftpd_dir_trav.nasl):

tftp/*/overflow (tftpd_overflow.nasl):

tftp/*/smalloverflow (tftpd_small_overflow.nasl):

/tmp/cgibin (DDI_Directory_Scanner.nasl):

/tmp/http/auth/* (http_login.nasl):

/tmp/hydra/empty_password (hydra_options.nasl):

/tmp/hydra/exit_ASAP (hydra_options.nasl):

/tmp/hydra/login_password (hydra_options.nasl):

/tmp/hydra/tasks (hydra_options.nasl):

/tmp/hydra/timeout (hydra_options.nasl):

/tmp/rpc/noportmap/* (misc_func.inc):

/tmp/UnableToRun/14254 (nikto.nasl):

/tmp/UnableToRun/14255 (nmap.nasl):

/tmp/UnableToRun/14663 (amap.nasl):

/tmp/UnableToRun/91984 (ldapsearch.nasl):

/tmp/UnableToRun/80000 (ike-scan.nasl):

Transport/SSL (find_service.nes):

webmin/*/version (webmin.nasl):

www/abyss (http_version.nasl):

www/akamaighost (http_version.nasl):

www/alchemy (http_version.nasl):

www/alibaba (http_version.nasl):

www/allegro (http_version.nasl):

www/anti-OpenVAS/*/rand-url (anti_nessus.nasl):

www/anti-OpenVAS/*/user-agent (anti_nessus.nasl):

www/anweb (http_version.nasl):

www/aolserver (http_version.nasl):

www/apache (http_version.nasl): 1 = Auf dem Zielrechner läuft Apache

www/appleshareip (http_version.nasl):

www/badblue (http_version.nasl):

www/banner/* (apache_SSL_complain.nasl, http_version.nasl):

www/BitKeeper (http_version.nasl):

www/buggy_post_crash (monkeyweb_post_DoS.nasl):

www/caudium (http_version.nasl):

www/cern (http_version.nasl): 1 = Auf dem Zielrechner läuft CERN httpd

www/communigatepro (http_version.nasl):

www/communique (http_version.nasl):

www/compaq (http_version.nasl):

www/cougar (http_version.nasl):

www/cups (http_version.nasl):

www/doc_browseable (doc_browsable.nasl):

www/domino (http_version.nasl): 1 = Auf dem Zielrechner läuft domino

www/domino/*/db (domino_default_db.nasl):

www/emweb (http_version.nasl):

www/filemakerpro (http_version.nasl):

www/firstclass (http_version.nasl):

www/goahead (http_version.nasl):

www/hmap/*/banner_ok (www_fingerprinting_hmap.nasl):

www/hmap/*/banner_regex (www_fingerprinting_hmap.nasl):

www/hmap/*/description (www_fingerprinting_hmap.nasl):

www/hmap/*/raw_signature (www_fingerprinting_hmap.nasl):

www/hmap/*/signature (www_fingerprinting_hmap.nasl):

www/ibm-http (http_version.nasl):

www/ideawebserver (http_version.nasl):

www/iis (http_version.nasl):

www/iplanet (http_version.nasl):

www/ipswitch-imail (http_version.nasl):

www/jetty (http_version.nasl):

www/jigsaw (http_version.nasl):

www/KFWebServer (http_version.nasl):

www/linuxconf (http_version.nasl):

www/miniserv (http_version.nasl):

www/multiple_get/* (www_multiple_get.nasl):

www/ncsa (http_version.nasl):

www/netcache (http_version.nasl):

www/netscape-commerce (http_version.nasl):

www/netscape-fasttrack (http_version.nasl):

www/netware (http_version.nasl):

www/novell (http_version.nasl):

www/omnihttpd (http_version.nasl):

www/OracleApache (http_version.nasl):

www/oracle-web-listener (http_version.nasl):

www/pi3web (http_version.nasl):

www/*/generic_xss (cross_site_scripting.nasl):

www/*/punBB (punBB_detect.nasl):
www/*/vBulletin (vbulletin_detect.nasl):
www/*/webmin (webmin.nasl):
www/real_banner/* (http_version.nasl):
www/resin (http_version.nasl):
www/roxen (http_version.nasl):
www/sambar (http_version.nasl):
www/savant (http_version.nasl):
www/simpleserver (http_version.nasl):
www/squid (http_version.nasl):
www/statistics-server (http_version.nasl):
www/stronghold (http_version.nasl):
www/stweb (http_version.nasl):
www/theserver (http_version.nasl):
www/thttpd (http_version.nasl):
www/tigershark (http_version.nasl):
www/tomcat (http_version.nasl):
www/too_big_post_crash (monkeyweb_too_big_post.nasl):
www/too_long_url_crash (oracle9iAS_too_long_url.nasl, ws4e_too_long_url.nasl):
www/tux (http_version.nasl):
www/visualroute (http_version.nasl):
www/vnc (vnc_http.nasl):
www/vqserver (http_version.nasl):
www/wdaemon (http_version.nasl):
www/weblogic (http_version.nasl):
www/webserver4everyone (http_version.nasl):
www/websitepro (http_version.nasl):
www/webstar (http_version.nasl):
www/wwwfileshare (http_version.nasl):
www/xitami (http_version.nasl):
www/zeus (http_version.nasl):
www/zope (http_version.nasl):
X11/*/answer (X.nasl):
X11/*/open (X.nasl):
X11/*/version (X.nasl):
xedus/*/running (xedus_detect.nasl):
zebra/banner/* (find_service2.nasl):
zebra/banner/* (zebra_dos.nasl):
zonealarm/version (zone_alarm_local_dos.nasl):

9.4 Tests und Fehlersuche

Es gibt verschiedene Arten, wie Sie Ihre OpenVAS NVTs testen können; beispielsweise unterscheiden sich netzwerkbasierte Tests von lokalen Sicherheitstests. Für lokale Sicherheitstests ist es unter Umständen wichtig, die Befehlszeile und die Ausgaben der vom Skript durchgeführten Aktionen zu sehen. Ein guter Ausgangspunkt ist das Programm `openvas-nasl`, mit dem Sie Ihre Skript in der Zielumgebung ausführen und auf Fehler überprüfen können.

9.4.1 Lokale Sicherheitstests

Sie können `openvas-nasl` wie folgt nutzen, um Ihre NVTs zu testen:

Stellen Sie zuerst sicher, dass Ihr NASL-Skript syntaktisch korrekt ist. Sie können dies in `openvas-nasl` mit der Option `-p` testen:

```
# openvas-nasl -p broken-example.nasl
syntax error, unexpected IDENT, expecting ')'
Parse error at or near line 17
```

Diese Meldung zeigt, dass in dem getesteten ein Syntaxfehler vorhanden ist. Um die Funktionalität Ihrer Skripts testen zu können, muss das Skript syntaktisch korrekt sein.

Nachdem Sie die Syntaxfehler beseitigt haben, können Sie Ihr Skript ausführen und die Ausgabe in eine Protokolldatei aufzeichnen:

```
openvas-nasl  -T /tmp/debug-lvt.txt -X  example-lvt.nasl
```

Die Ausgaben des Skripts werden in die Datei `debug-lvt.txt` geschrieben, die wie folgt aussehen könnte:

```
[...]
NASL:0196> make_list(...)
[9831]() NASL> [080c9968] <- "qpkg"
[9831]() NASL> [080c9a00] <- "-nc"
[9831]() NASL> [080c9f38] <- "-I"
[9831]() NASL> [080c9f98] <- "-v"
[9831](example-lvt.nasl) NASL> Call make_list(1: "qpkg", 2: "-nc", 3: "-I", 4:
"-v")
[9831](example-lvt.nasl) NASL> Return make_list: ???? (DYN_ARRAY (64))
[9831]() NASL> [080c9e88] <- (VAR2_ARRAY)
[9831](example-lvt.nasl) NASL> Call pread(cmd: "qpkg", argv: ???? (DYN_ARRAY
(64)))
[9831](example-lvt.nasl) NASL> Return pread: "qpkg: invalid option -- n
Usage: qpkg <opt..."
[9831]() NASL> [080c95c0] <- "qpkg: invalid option -- n
Usage: qpkg <opts> <misc args> : manipulate Gentoo binpkgs

Options: -[cpP:vqChV]
  -c, --clean          * clean pkgdir of unused binary files
  -p, --pretend        * pretend only
  -P, --pkgdir   <arg> * alternate package directory
  -v, --verbose        * Make a lot of noise
  -q, --quiet          * Tighter output; suppress warnings
  -C, --nocolor        * Don't output color
  -h, --help           * Print this help and exit
  -V, --version        * Print version and exit
"
NASL:0199> if (! (qpkg_list)) { ... }
```

```
[9831](example-lvt.nasl) NASL> [080c95c0] -> "qpkg: invalid option -- n
Usage: qpkg <opts> <misc args> : manipulate Gentoo binpkgs

Options: -[cpP:vqChV]
  -c, --clean          * clean pkgdir of unused binary files
  -p, --pretend        * pretend only
  -P, --pkgdir   <arg> * alternate package directory
  -v, --verbose        * Make a lot of noise
  -q, --quiet          * Tighter output; suppress warnings
  -C, --nocolor        * Don't output color
  -h, --help           * Print this help and exit
  -V, --version        * Print version and exit
"
NASL:0201> if (((arch) && (my_arch)) && (my_arch >!< arch)) { ... }
[9831](example-lvt.nasl) NASL> [080c9948] -> undef
NASL:0201> l=egrep(...);
NASL:0201> egrep(...)
[9831](example-lvt.nasl) NASL> [080c95c0] -> "qpkg: invalid option -- n
Usage: qpkg <opts> <misc args> : manipulate Gentoo binpkgs

Options: -[cpP:vqChV]
  -c, --clean          * clean pkgdir of unused binary files
  -p, --pretend        * pretend only
  -P, --pkgdir   <arg> * alternate package directory
  -v, --verbose        * Make a lot of noise
  -q, --quiet          * Tighter output; suppress warnings
  -C, --nocolor        * Don't output color
  -h, --help           * Print this help and exit
  -V, --version        * Print version and exit
"
[9831]() NASL> [080c9890] <- "qpkg: invalid option ? n
[...]
```

Die letzte Zeile zeigt an, dass eine inkorrekter Syntax für den Aufruf des Programms `qpkg` verwendet wurde.

9.4.2 Netzwerk-Sicherheitstests

Sie können `openvas-nasl` wie folgt nutzen, um Ihre NVTs zu testen:

Stellen Sie zuerst sicher, dass Ihr NASL-Skript syntaktisch korrekt ist. Sie können dies in `openvas-nasl` mit der Option `-p` testen:

```
# openvas-nasl -p broken-example.nasl
syntax error, unexpected IDENT, expecting ')'
Parse error at or near line 17
```

Diese Meldung zeigt, dass in dem getesteten ein Syntaxfehler vorhanden ist. Um die Funktionalität Ihrer Skripts testen zu können, muss das Skript syntaktisch korrekt sein.

Das Testen eines NVTs, das einen Test über das Netzwerk ausführt, ist etwas aufwändiger. Um zu überprüfen, ob das richtige Paket gesendet wurde, könne Sie beispielsweise das Programm `tcpdump` benutzen, um die Kommunikation zwischen Ihrem System und dem Zielsystem zu protokollieren:

```
# tcpdump -i lo -w debug.pcap -s 1450
tcpdump: listening on lo, link-type EN10MB (Ethernet), capture size 1450 bytes
10 packets captured
20 packets received by filter
0 packets dropped by kernel
```

Nun können Sie sich den Inhalt der Netzwerkkommunikation anzeigen lassen:

```
 # tcpdump -vvvv -n -r debug.pcap
reading from file debug.pcap, link-type EN10MB (Ethernet)
15:45:52.474613 IP (tos 0x0, ttl 64, id 60969, offset 0, flags [DF], proto TCP
(6), length 60) 127.0.0.1.53655 > 127.0.0.1.24: S, cksum 0x80c9 (correct),
1315997236:1315997236(0) win 32792 <mss 16396,sackOK,timestamp 5466141
0,nop,wscale 6>
15:45:52.474618 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6),
length 60) 127.0.0.1.24 > 127.0.0.1.53655: S, cksum 0x64b5 (correct),
1311860089:1311860089(0) ack 1315997237 win 32768 <mss 16396,sackOK,timestamp
5466141 5466141,nop,wscale 6>
15:45:52.474638 IP (tos 0x0, ttl 64, id 60970, offset 0, flags [DF], proto TCP
(6), length 52) 127.0.0.1.53655 > 127.0.0.1.24: ., cksum 0x4bd8 (correct),
1:1(0) ack 1 win 513 <nop,nop,timestamp 5466141 5466141>
15:45:52.474797 IP (tos 0x0, ttl 64, id 3431, offset 0, flags [DF], proto TCP
(6), length 72) 127.0.0.1.24 > 127.0.0.1.53655: P, cksum 0xfe3c (incorrect (->
0x941e), 1:21(20) ack 1 win 512 <nop,nop,timestamp 5466141 5466141>
15:45:52.474829 IP (tos 0x0, ttl 64, id 60971, offset 0, flags [DF], proto TCP
(6), length 52) 127.0.0.1.53655 > 127.0.0.1.24: ., cksum 0x4bc3 (correct),
1:1(0) ack 21 win 513 <nop,nop,timestamp 5466142 5466141>
15:45:52.475572 IP (tos 0x0, ttl 64, id 60972, offset 0, flags [DF], proto TCP
(6), length 68) 127.0.0.1.53655 > 127.0.0.1.24: P, cksum 0xfe38 (incorrect (->
0xefa4), 1:17(16) ack 21 win 513 <nop,nop,timestamp 5466142 5466141>
15:45:52.475586 IP (tos 0x0, ttl 64, id 3432, offset 0, flags [DF], proto TCP
(6), length 52) 127.0.0.1.24 > 127.0.0.1.53655: ., cksum 0x4bb3 (correct),
21:21(0) ack 17 win 512 <nop,nop,timestamp 5466142 5466142>
15:45:57.479223 IP (tos 0x0, ttl 64, id 60973, offset 0, flags [DF], proto TCP
(6), length 52) 127.0.0.1.53655 > 127.0.0.1.24: F, cksum 0x46ce (correct),
17:17(0) ack 21 win 513 <nop,nop,timestamp 5467393 5466142>
15:45:57.479279 IP (tos 0x0, ttl 64, id 3433, offset 0, flags [DF], proto TCP
(6), length 52) 127.0.0.1.24 > 127.0.0.1.53655: F, cksum 0x41eb (correct),
21:21(0) ack 18 win 512 <nop,nop,timestamp 5467393 5467393>
15:45:57.479296 IP (tos 0x0, ttl 64, id 60974, offset 0, flags [DF], proto TCP
(6), length 52) 127.0.0.1.53655 > 127.0.0.1.24: ., cksum 0x41ea (correct),
18:18(0) ack 22 win 513 <nop,nop,timestamp 5467393 5467393>
```

Sollte eine tiefergehende Analyse notwendig sein, können Programme wie `wireshark` Dateien im `pcap`-Format importieren und Ihnen bei der Fehlersuche behilflich sein.

Der `openvas-nasl` Interpreter kann wie oben beschrieben zusätzlich noch eine Protokolldatei erstellen. Diese kann auch bei der Fehlersuche in netzwerkbasierten Sicherheitstests hilfreich sein:

```
[...]
NASL:0277> register_int_in_kb(...)
[9905](ssh_detect24.nasl) NASL> [0811e310] -> 0
[9905]() NASL> [08120328] <- 0
[9905]() NASL> [08120358] <- "Secret/SSH/bugged_sshd"
[9905](ssh_detect24.nasl) NASL> Call register_int_in_kb(int: 0, name:
"Secret/SSH/bugged_sshd")
NASL:0055> if ((! (defined_func(...))) || (! (_reuse_connection))) { ... }
NASL:0054> defined_func(...)
[9905]() NASL> [081203a8] <- "replace_kb_item"
[9905](ssh_detect24.nasl) NASL> Call defined_func(1: "replace_kb_item")
[9905](ssh_detect24.nasl) NASL> Return defined_func: 1
[9905](ssh_detect24.nasl) NASL> [0811e2d8] -> undef
NASL:0054> return 0;
[9905](ssh_detect24.nasl) NASL> Return register_int_in_kb: 0
[9905](ssh_detect24.nasl) NASL> Return init: FAKE
NASL:1771> server_version=ssh_exchange_identification(...);
NASL:1771> ssh_exchange_identification(...)
[9905](ssh_detect24.nasl) NASL> [0811fde0] -> 1000000
[9905]() NASL> [08120688] <- 1000000
[9905](ssh_detect24.nasl) NASL> Call ssh_exchange_identification(socket:
1000000)
NASL:0377> local_var ...
```

```
NASL:0379> buf=recv_line(...);
NASL:0379> recv_line(...)
[9905](ssh_detect24.nasl) NASL> [08120688] -> 1000000
[9905]() NASL> [081207b0] <- 1000000
[9905]() NASL> [081207d0] <- 1024
[9905](ssh_detect24.nasl) NASL> Call recv_line(socket: 1000000, length: 1024)
[9905](ssh_detect24.nasl) NASL> Return recv_line: "SSH-2.0-FreeSSH_9.9
"
[9905]() NASL> [081202d0] <- "SSH-2.0-FreeSSH_9.9
"
NASL:0388> if (! (buf)) { ... }
[9905](ssh_detect24.nasl) NASL> [081202d0] -> "SSH-2.0-FreeSSH_9.9
"
NASL:0394> if (! (ereg(...))) { ... }
NASL:0388> ereg(...)
[9905](ssh_detect24.nasl) NASL> [081202d0] -> "SSH-2.0-FreeSSH_9.9
"
[9905]() NASL> [081206a8] <- "SSH-2.0-FreeSSH_9.9
"
[9905]() NASL> [081207b0] <- "^SSH-*[0-9]\.*[0-9]-*[^\n]"
[9905](ssh_detect24.nasl) NASL> Call ereg(string: "SSH-2.0-FreeSSH_9.9
", pattern: "^SSH-*[0-9]\.*[0-9]-*[^\n]")
[9905](ssh_detect24.nasl) NASL> Return ereg: 1
NASL:0394> sshversion=split(...);
NASL:0394> split(...)
[9905](ssh_detect24.nasl) NASL> [081202d0] -> "SSH-2.0-FreeSSH_9.9
"
[9905]() NASL> [08120638] <- "SSH-2.0-FreeSSH_9.9
"
[9905]() NASL> [081207b0] <- "-"
[9905]() NASL> [0811fff8] <- 0
[9905](ssh_detect24.nasl) NASL> Call split(1: "SSH-2.0-FreeSSH_9.9
", sep: "-", keep: 0)
[9905](ssh_detect24.nasl) NASL> Return split: ???? (DYN_ARRAY (64))
[9905]() NASL> [081202f0] <- (VAR2_ARRAY)
NASL:0395> num=split(...);
NASL:0395> split(...)
[...]
```

Diese Informationen sollten ausreichen, um das Problem zu lösen. Falls nicht, könnte der Fehler unter Umständen auch in der Skriptausführung von OpenVAS liegen. Dies können Sie überprüfen, indem Sie den NASL Interpreter mit Debugsymbolen kompilieren und einen Debugger wie `gdb` benutzen. Nähere Informationen über `gdb` finden Sie unter: http://www.gnu.org/software/gdb/gdb.html.

9.5 SMBclient-basierte WLSC NASL Skripte schreiben

(von Carsten Koch Mauthe)

Die API für die Benutzung von SMBclient wird durch die Funktionsbibliothek `smbcl_func.inc` zur Verfügung gestellt. Diese Datei muss in jedes WLSC eingebunden werden.

smbclientavail()

Diese Funktion gibt den Wert `TRUE` zurück, wenn `smbclient` aus OpenVAS heraus ausgeführt werden kann. Sie erstellt auch den „SMB/smbclient“-Eintrag in der Wissensbasis und sollte zu Beginn eines jeden WLSC-Skripts aufgerufen werden. Beispiel:

```
include("smbcl_func.inc");
```

```
if(!get_kb_item("SMB/smbclient"))
{
  smbclientavail();
}
if(get_kb_item("SMB/smbclient"))
{
  do something;
}
else
{
  report = string("SMBClient not found on this host !");
  security_note(port:0, proto:"SMBClient", data:report);
  exit(0);
}
```

smbversion()

Diese Funktion liefert den Wert `TRUE` zurück und schreibt die Domäne, die Version des Betriebssystems und die Version des Samba-Servers in die Wissensbasis unter den Einträgen `SMB/DOMAIN`, `SMB/OS` und `SMB/SERVER`.

smbgetfile(share, filename, tmp_filename)

Diese Funktion kann genutzt werden, um eine Datei vom Zielrechner herunterzuladen und sie lokal unter dem Namen `tmp_filename` zu speichern. Sie liefert bei Erfolg den Wert `TRUE`zurück. Beispiel:

```
tmp_filename = get_tmp_dir() + "tmpfile" + rand();
orig_filename = "C:\Windows\systems32\ntdll.dll";
smbgetfile(share: "C$", filename: orig_filename, tmp_filename: tmp_filename)
```

smbgetdir(share, dir, typ)

Diese Funktion kann genutzt werden, um Verzeichniseinträge aus der SMB-Quelle auszulesen.

typ = 0: Alle Einträge

typ = 1: Nur Dateien

typ = 2: Nur Verzeichnisse

Damit ist es möglich, auf eine oder mehrere Dateien oder Verzeichnisse zu überprüfen. Die Funktion liefert den Wert `NULL` oder ein Array der gefundenen Einträge zurück. Beispiel:

```
r = smbgetdir(share: "C$", dir: "C:\Windows\systems32\*.dll", typ: 1);
```

GetPEFileVersion (tmp_filename, orig_filename)

Diese Funktion liefert die Version von ausführbaren Dateien vom Typ „Windows PE/32“ (wie etwa „.exe“ oder „.dll“) zurück. Dies kann zusammen mit `smbgetfile` dazu genutzt werden, um Windows-Sicherheitslücken zu identifizieren.

Beispiel:

```
  tmp_filename = get_tmp_dir() + "tmpfile" + rand();
  orig_filename = "C:\Windows\systems32\ntdll.dll";
  if(smbgetfile(share: "C$", filename: orig_filename,
                tmp_filename: tmp_filename))
  {
    v = GetPEFileVersion(tmp_filename:tmp_filename,
                         orig_filename:orig_filename);
    if(v = "1.2.3.4")
    {
      ...
```

get_windir()

Diese Funktion gibt das Windows-Verzeichnis `WINNT` oder `WINDOWS` zurück, abhängig von dem durch `smbversion` identifizierten Betriebssystem.

9.5.1 Beispiel

Dies ist ein vollständiges NASL-Skript für einen WLSC. Es kann in den OpenVAS-NVTs unter dem Namen `win_CVE-2007-0043.nasl` gefunden werden.

```
#
# This script was written by
# Carsten Koch-Mauthe
# <c.koch-mauthe at dn-systems.de>
#
# This script is released under the GNU GPLv2
#
# $Revision: 01 $

if(description)
{
  if (OPENVAS_NASL_LEVEL >= 2206)
  {
    script_oid("1.3.6.1.4.1.25623.1.0.90010");
  }
  else
  {
    script_id(90010);
  }
  script_version ("$Revision: 01 $");
  script_cve_id("CVE-2007-0043");
  name["english"] = ".NET JIT Compiler Vulnerability";
  script_name(english:name["english"]);

  desc["english"] =
"The remote host is affected by the vulnerabilities described in CVE-2007-0043

Checking if System.web.dll version is less than 2.0.50727.832

Impact
    The Just In Time (JIT) Compiler service in Microsoft
    .NET Framework 1.0, 1.1, and 2.0 for Windows 2000, XP,
    Server 2003, and Vista allows user-assisted remote
    attackers to execute arbitrary code via unspecified
    vectors involving an unchecked buffer, probably a
    buffer overflow, aka .NET JIT Compiler Vulnerability.
    Checking if System.web.dll version is less than 2.0.50727.832

References:
    http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-0043
```

```
Solution:
    All users should upgrade to the latest version.


Risk factor : High";

  script_description(english:desc["english"]);
  summary["english"] = "Test for .NET JIT Compiler Vulnerability";
  script_summary(english:summary["english"]);
  script_category(ACT_GATHER_INFO);
  script_copyright(english:"This script is under GPLv2");
  family["english"] = "Windows.NET";
  script_family(english:family["english"]);
  exit(0);
}

#
# The code starts here
#

include("version_func.inc");
include("smbcl_func.inc");

if(!get_kb_item("SMB/smbclient"))
{
  smbclientavail();
}
test_version = "2.0.50727.832";

if(get_kb_item("SMB/smbclient"))
{
  if(smbversion() == 0)
  {
    report = string("Error getting SMB-Data -> " +
                    get_kb_item("SMB/ERROR"));
    security_note(port:0, proto:"SMBClient", data:report);
    exit(0);
  }
}
else
{
  report = string("SMBClient not found on this host !");
  security_note(port:0, proto:"SMBClient", data:report);
  exit(0);
}

win_dir = get_windir();
if(!isnull(win_dir))
{
  path = win_dir+"Microsoft.NET\Framework\";
  filespec = "v2*";
  r = smbgetdir(share: "C$", dir: path+filespec, typ: 2);
  if(!isnull(r))
  {
    filespec = r[0] + "\" + "system.web.dll";
    r = smbgetdir(share: "C$", dir: path+filespec, typ: 1);
    if(!isnull(r))
    {
      tmp_filename = get_tmp_dir() + "tmpfile" + rand();
      orig_filename = path + filespec;
      if(smbgetfile(share: "C$", filename: orig_filename,
                    tmp_filename: tmp_filename))
      {
        v = GetPEFileVersion(tmp_filename:tmp_filename,
                             orig_filename:orig_filename);
```

```
        unlink(tmp_filename);
        if(version_is_less(version: v, test_version: test_version))
        {
          security_hole(port:0, proto:"SMB");
          report = report + "Fileversion : C$ " +
                    orig_filename + " "+ v + string("\n");
          security_hole(port:0, proto:"SMB", data:report);
        }
      }
      else
      {
        report = string("Error getting SMB-File -> " +
                        get_kb_item("SMB/ERROR")) + string("\n");
        security_note(port:0, proto:"SMB", data:report);
      }
    }
  }
  else
  {
    report = string(".NET V2xx not found/no access -> " +
                    get_kb_item("SMB/ERROR")) + string("\n");
    security_note(port:0, proto:"SMB", data:report);
  }
}

exit(0);
```

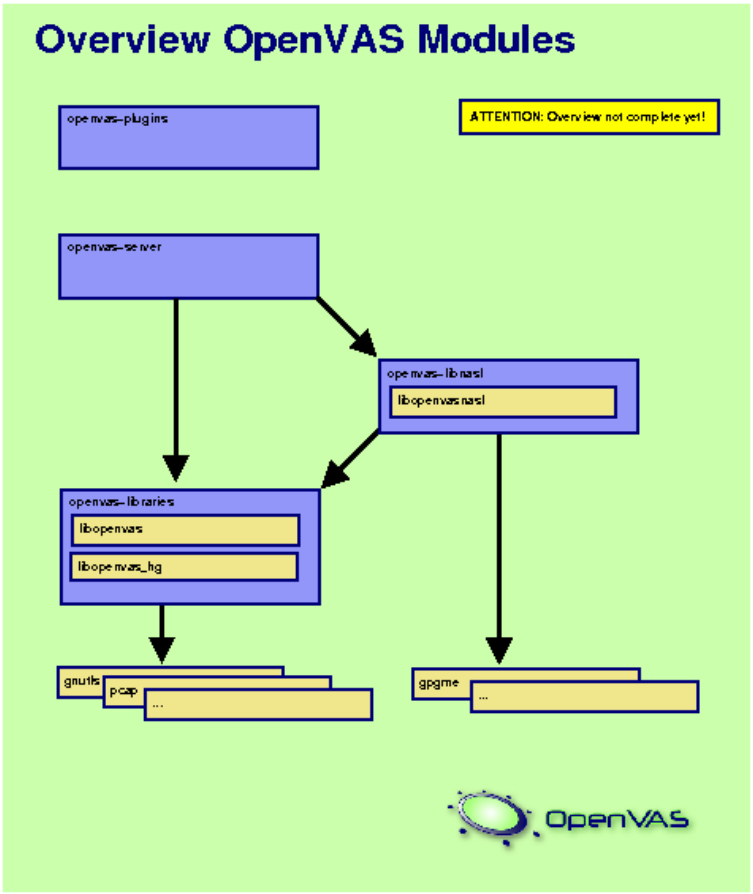
10 OpenVAS-Entwicklerhandbuch

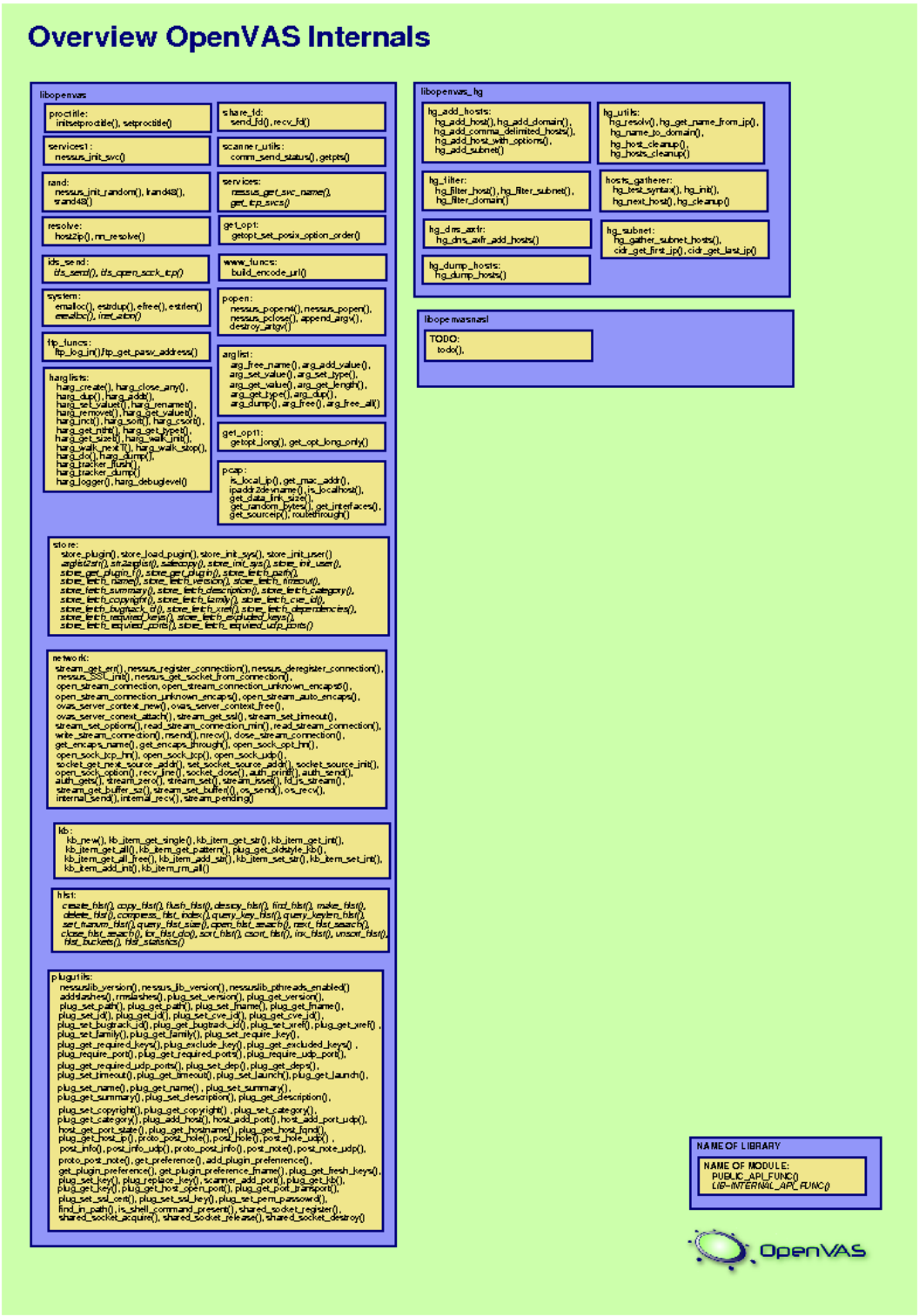
(von Jan-Oliver Wagner)

10.1 Die OpenVAS Quelltext-Karte

Ein großer Teil der Codebasis von OpenVAS wurde von Nessus übernommen. Wie bei allen Projekten dieser Größenordnung kann es zu Beginn für neue Projektteilnehmer recht schwierig sein, sich in neuem, noch unbekannten Quelltext zurechtzufinden. Obwohl der Nessus-Quelltext an vielen Stellen sehr strukturiert ist, ist er an anderen Stellen eher chaotisch und nicht leicht zu verstehen. Um die Codebasis besser visualisieren zu können hat das OpenVAS-Projekt damit begonnen, Karten des Quelltextes und der Struktur zu erstellen.

Im Folgenden finden Sie einen ersten Entwurf für die Übersicht und die interne Struktur. Berücksichtigen Sie bitte, dass diese Karten momentan überarbeitet werden; aktuellere Karten sind meist über die OpenVAS-Website verfügbar.




Overview OpenVAS Internals
libopenvas
proctitle:
initsetproctitle(), setproctitle()
services1:
nessus_init_svc()
rand:
nessus_init_random(), krand48(), srand48()
resolve:
host2ip(), rm_resolve()
kb_send:
kb_send(), kb_open_sock_kb()
system:
emalloc(), estrdup(), efree(), estrlen()
emalloc(), inet_aton()
ftp_funcs:
ftp_log_in()ftp_get_pasv_address()
harg lists:
harg_create(), harg_close_any(),
harg_dup(), harg_add(),
harg_set_value(), harg_rename(),
harg_removed(), harg_get_value(),
harg_incr(), harg_sort(), harg_csort(),
harg_get_nthi(), harg_get_type(),
harg_get_sizet(), harg_walk_init(),
harg_walk_next(), harg_walk_stop(),
harg_dec(), harg_dump(),
harg_tracker_flush()
harg_tracker_dump()
harg_logger(), harg_debuglevel()
share_fd:
send_fd(), recv_fd()
scanner_utils:
comm_send_status(), getpts()
services:
nessus_get_svc_name(),
get_tcp_svcs()
get_opt:
getopt_set_posix_option_order()
www_funcs:
build_encode_url()
popen:
nessus_popen4(), nessus_popen(),
nessus_pclose(), append_argv(),
destroy_argv()
arg list:
arg_free_name(), arg_add_value(),
arg_set_value(), arg_set_type(),
arg_get_value(), arg_get_length(),
arg_get_type(), arg_dup(),
arg_dump(), arg_free(), arg_free_all()
get_opt1:
getopt_long(), get_opt_long_only()
pcap:
is_local_ip(), get_mac_addr(),
ipaddr2devname(), is_localhost(),
get_data_link_size(),
get_random_bytes(), get_interfaces(),
get_source_ip(), routethrough()
store:
store_plugin(), store_load_plugin(), store_init_sys(), store_init_user()
arglist2str(), str2arglist(), safeempty(), store_init_sys(), store_init_user(),
store_get_plugin_fl(), store_get_plugin(), store_fetch_path(),
store_fetch_name(), store_fetch_version(), store_fetch_timeout(),
store_fetch_summary(), store_fetch_description(), store_fetch_category(),
store_fetch_copyright(), store_fetch_family(), store_fetch_cve_id(),
store_fetch_bugtrack_id(), store_fetch_xref(), store_fetch_dependencies(),
store_fetch_required_keys(), store_fetch_excluded_keys(),
store_fetch_required_ports(), store_fetch_required_udp_ports()
network:
stream_get_err(), nessus_register_connection(), nessus_deregister_connection(),
nessus_SSL_init(), nessus_get_socket_from_connection(),
open_stream_connection, open_stream_connection_unknown_encaps(),
open_stream_connection_unknown_encaps(), open_stream_auto_encaps(),
ovas_server_context_new(), ovas_server_context_free(),
ovas_server_conext_attach(), stream_get_ssl(), stream_set_timeout(),
stream_set_options(), read_stream_connection_min(), read_stream_connection(),
write_stream_connection(), nsend(), nrecv(), close_stream_connection(),
get_encaps_name(), get_encaps_through(), open_sock_opt_hn(),
open_sock_tcp_hn(), open_sock_tcp(), open_sock_udp(),
socket_get_next_source_addr(), set_socket_source_addr(), socket_source_init(),
open_sock_option(), recv_line(), socket_close(), auth_printf(), auth_send(),
auth_gets(), stream_zero(), stream_set(), stream_nsend(), fd_is_stream(),
stream_get_buffer_size(), stream_set_buffer(), os_send(), os_recv(),
internal_send(), internal_recv(), stream_pending()
kb:
kb_new(), kb_item_get_single(), kb_item_get_str(), kb_item_get_int(),
kb_item_get_all(), kb_item_get_pattern(), plug_get_old_style_kb(),
kb_item_get_all_free(), kb_item_add_str(), kb_item_set_str(), kb_item_set_int(),
kb_item_add_int(), kb_item_rm_all()
hlist:
create_hlist(), copy_hlist(), flush_hlist(), destroy_hlist(), first_hlist(), make_hlist(),
delete_hlist(), compress_hlist_nodes(), query_key_hlist(), query_key_len_hlist(),
set_maxnum_hlist(), query_hlist_size(), open_hlist_search(), next_hlist_search(),
close_hlist_search(), for_hlist_do(), sort_hlist(), csort_hlist(), ins_hlist(), unsort_hlist(),
hlist_buckets(), hlist_statistics()
plug utils:
nessuslib_version(), nessus_lib_version(), nessuslib_pthreads_enabled()
addslashes(), rmslashes(), plug_set_version(), plug_get_version(),
plug_set_path(), plug_get_path(), plug_set_fname(), plug_get_fname(),
plug_set_id(), plug_get_id(), plug_set_cve_id(), plug_get_cve_id(),
plug_set_bugtrack_id(), plug_get_bugtrack_id(), plug_set_xref(), plug_get_xref() ,
plug_set_family(), plug_get_family(), plug_set_require_key(),
plug_get_required_keys(), plug_exclude_key(), plug_get_excluded_keys() ,
plug_require_port(), plug_get_required_ports(), plug_require_udp_port(),
plug_get_required_udp_ports(), plug_set_dep(), plug_get_deps(),
plug_set_timeout(), plug_get_timeout(), plug_set_launch(), plug_get_launch(),
plug_set_name(), plug_get_name() , plug_set_summary(),
plug_get_summary(), plug_set_description(), plug_get_description(),
plug_set_copyright(), plug_get_copyright() , plug_set_category(),
plug_get_category(), plug_add_host(), host_add_port(), host_add_port_udp(),
host_get_port_state(), plug_get_hostname(), plug_get_host_fqdn(),
plug_get_host_ip(), proto_post_hole(), post_hole(), post_hole_udp() ,
post_info(), post_info_udp(), proto_post_info(), post_note(), post_note_udp(),
proto_post_note(), get_preference(), add_plugin_preference(),
get_plugin_preference(), get_plugin_preference_fname(), plug_get_fresh_keys(),
plug_set_key(), plug_replace_key(), scanner_add_port(), plug_get_kb(),
plug_get_key(), plug_get_host_open_port(), plug_get_port_transport(),
plug_set_ssl_cert(), plug_set_ssl_key(), plug_set_pem_password(),
find_in_path(), is_shell_command_present(), shared_socket_register(),
shared_socket_acquire(), shared_socket_release(), shared_socket_destroy()
libopenvas_hg
hg_add_hosts:
hg_add_host(), hg_add_domain(),
hg_add_comma_delimited_hosts(),
hg_add_host_with_options(),
hg_add_subnet()
hg_utils:
hg_resolv(), hg_get_name_from_ip(),
hg_name_to_domain(),
hg_host_cleanup(),
hg_hosts_cleanup()
hg_filter:
hg_filter_host(), hg_filter_subnet(),
hg_filter_domain()
hosts_gatherer:
hg_test_syntax(), hg_init(),
hg_next_host(), hg_cleanup()
hg_dns_axfr:
hg_dns_axfr_add_hosts()
hg_subnet:
hg_gather_subnet_hosts(),
cidr_get_first_ip(), cidr_get_last_ip()
hg_dump_hosts:
hg_dump_hosts()
libopenvasnasl
TODO:
todo(),
NAME OF LIBRARY
NAME OF MODULE:
PUBLIC_API_FUNC()
LIB-INTERNAL_API_FUNC()
OpenVAS

10.2 Quelltext-Zweige für stabile und Entwicklungsversionen

Wenn Sie den OpenVAS-Quelltext zum ersten Mal anschauen, fragen Sie sich wahrscheinlich, wozu all diese „branches“ (Zweige) und „tags“ (Markierungen) gut sind und warum die meisten Änderungen im „trunk“ (Stamm) passieren.

Wie viele andere Projekte nutzt OpenVAS das Konzept von Zweigen und Markierungen, um wichtige Schritte in der Quelltextverwaltung festzuhalten. Wie Sie eventuell bereits bemerkt haben, finden so gut wie alle Änderungen am Quelltext im Stamm statt. Der Stamm ist der Ort, wo Fehler beseitigt werden und neue Funktionalität zuerst erscheint. [1]

Wenn ein Module sich stabilisiert hat und die Änderungen an diesem Modul von anderen Entwicklern geprüft wurden, findet eine Entscheidung über ein neues Release statt. Die Absicht, eine neue Version zu veröffentlichen, wird im Allgemeinen auf der Mailingliste der Entwickler geäußert; wenn sich dazu eine Zustimmung findet, wird ein Release vorbereitet. Als Teil des Releaseablaufs wird die aktuelle Revision des Moduls markiert („tagged“) und erscheint unter den Tags mit einer neuen Versionsnummer, beispielsweise `openvas-client-release-2.1.0`.

Eine markierte Version ist eine Momentaufnahme des Quelltextes zum Zeitpunkt des Releases; dieser erlaubt es unter anderem Benutzern, alte Versionen problemlos aus der Quelltextverwaltung abzurufen, falls dies notwendig werden sollte. Eine markierte Version wird nicht mehr weiterentwickelt. Sollten Änderungen an einer markierten Version notwendig werden, werden diese Änderungen im Stamm oder in dem ursprünglichen Zweig des Moduls vorgenommen und ein neues Release mit einer neuen Versionsnummer erstellt.

Wie Sie anhand des letzten Satzes vermuten können, unterscheiden sich Zweige und Markierungen dadurch, dass in einem Zweig Weiterentwicklungen stattfinden können. Der Quelltext wird in der Regel verzweigt, wenn größere Änderungen anstehen, die den Stamm für eine gewisse Zeit instabil machen werden oder mit älteren Versionen nicht mehr kompatibel sein werden. Ein Grund für das Verzweigen ist, dass so größere Änderung möglich sind und gleichzeitig eine getrennter Zweig erhalten bleibt, indem rasch kleinere Fehler an bereits veröffentlichten Versionen vorgenommen werden können.

Beispielsweise könnte nach der Veröffentlichung der Version 2.1.5 des Moduls `openvas-client` der Beschluss gefasst werden, dass bevorstehende größere Änderungen eine Verzweigung rechtfertigen. Dies führt dazu, dass eine neuer Zweig mit dem Namen `openvas-client-2-1` erstellt wird. Weitere Releases aus der 2.1er-Serie (2.1.6, 2.1.7) werden aus diesem Zweig erstellt, während die größeren Änderungen im Stamm letztlich in der Veröffentlichung der Version 2.2.0 münden werden.

Falls Fehler im Quelltext gefunden werden, die sowohl den Stamm als auch einen oder mehrere Zweige betreffen, werden diese Fehler im Allgemeinen zuerst im Stamm behoben und dann in die einzelnen Zweige übertragen. Die Übertragung für als „backporting“ bezeichnet. Diese „Backports“ stellen den Großteil der Änderungen dar, die in einem Zweig geschehen. Es sollte nur äußert selten notwendig sein, eine Änderung in die Gegenrichtung zu übertragen, also Änderungen in einem Zweig in den Stamm zurückzuführen. Entwickler sollten es nach Möglichkeit vermeiden, Änderungen an einem Zweig vorzunehmen die noch nicht am Stamm durchgeführt wurden, solange kein zwingender Grund dafür besteht.

10.3 Codequalität und Codesicherheit

Besonders als Hersteller eines Produktes für die IT-Sicherheit fühlt sich das OpenVAS-Projekt dazu verpflichtet, einen möglichst hohen Grad in Bezug auf die Codequalität und -sicherheit anzustreben. Obwohl der von Nessus übernommene Quelltext sicher in beiden Aspekten verbesserungswürdig war, vertraut das Projekt auf

[1] Auch wenn Sie gerne den Stamm benutzen würden, um die aktuellsten Änderungen und die neueste Funktionalität zu erhalten, sollten Sie bedenken, dass der Stamm ständig in Entwicklung begriffen ist und sich an Entwickler richtet. Dies bedeutet, dass davon abgeraten wird, den Stamm in Produktivumgebungen zu benutzen.

die Fähigkeiten und die Erfahrung der Teilnehmer um diese Unzulänglichkeiten zu beseitigen und Codequalität und -sicherheit weiter zu verbessern.

Das OpenVAS-Projekt nutzt verschiedene automatisierte Werkzeuge, um die Codequalität zu messen und potentielle Probleme im Quelltext zu identifizieren. Dazu werden unter anderem die Programme „Flawfinder“ und „RATS“ („Rough Auditing Tool for Security“) genutzt.

Die aktuellen Ergebnisse dieser Tests sind verfügbar unter:

http://www.openvas.org/code-quality.html

Bitte berücksichtigen Sie, dass Sie sich zunächst mit den verwendeten Programmen beschäftigen sollten, bevor Sie die absoluten Zahlen auswerten. Auch wenn jede automatische Meldung von den OpenVAS-Entwicklern überprüft wird, haben einige Meldungen keinen relevanten Einfluss auf die Codequalität und -sicherheit.

10.4 OpenVAS Change Requests

OpenVAS Change Requests (etwa: „Änderungswünsche“) beschreiben vorgeschlagene Änderungen an einem oder mehreren OpenVAS-Modulen. Obwohl dieser Ablauf formalisiert erscheinen mag, ist er kein Ersatz für offene Diskussionen unter interessierten Entwicklern auf den Mailinglisten. In der Tat haben viele Change Requests ihren Ursprung in solchen Diskussionen. Dieser Ansatz soll den Ablauf der OpenVAS-Entwicklung so transparent wie möglich machen, sowohl für die OpenVAS-Entwickler als auch für OpenVAS-Anwender und andere Interessenten.

Sie können die aktuellen Change Requests auf der OpenVAS-Website einsehen:

http://www.openvas.org/openvas-crs.html.

Ein vollständiger Change Request ist in englisch verfasst und enthält die folgenden Abschnitte:

Status: Eine allgemeinen Beschreibung des derzeitigen Standes dieses Change Requests, beispielsweise „in discussion“ (in Diskussion), „agreed (voted +3) for release 1.4“ (für Version 1.4 mit +3 Stimmen angenommen) oder „Step 1 and 2 implemented“ (Schritt 1 und 2 implementiert).

Purpose: Eine kurze Beschreibung der Gründe für diesen Change Request.

References: Links zu Beiträgen in Issue-Trackern oder Mailinglisten, auf die sich dieser Change Request bezieht.

Rationale: Eine ausführlichere Erklärung, warum dieser Change Request implementiert werden sollte.

Effects: Die Effekte, falls dieser Change Request implementiert würden, in Bezug auf das API, Kompatibilität, Benutzungsoberfläche usw.

Design and Implementation: Technische Einzelheiten zur Implementierung diese Change Requests.

History: Datum, Name und Beschreibung der Änderungen an diesem Change Request im „ChangeLog“-Format.

Change Requests können von jedem verfasst werden; obwohl die meisten Change Requests von OpenVAS-Entwicklern stammen, soll dies in keiner Weise andere davon abhalten, selbst Change Requests zu erstellen und sie an die Mailingliste „openvas-discuss“ zu senden. Autoren von Change Requests sollten zunächst ihre Ideen auf dieser Mailingliste oder im Online-Chat (#openvas auf irc.oftc.net) beschreiben, bevor sie mit der Zusammenstellung des Requests beginnen.

Die OpenVAS-Entwickler stimmen regelmäßig über neue Change Requests ab; jeder OpenVAS-Entwickler kann für oder gegen einen entsprechenden Change Request stimmen, im Allgemeinen mit den Werten „+1“ (dafür), „+/-0“ (etwas dafür/dagegen) oder „-1“ (dagegen). Nach einigen Tagen werden die Stimmen gezählt;

ein positives Ergebnis bedeutet, dass dieser Change Request akzeptiert wurde und in zukünftigen Versionen von OpenVAS implementiert wird.

Der Ablauf der Abstimmung ist noch nicht vollständig formalisiert und kann sich ändern; Ideen oder Anregungen sind herzlich willkommen.

10.5 Patches erstellen

Falls Sie einen Fehler im OpenVAS-Quelltext gefunden (und beseitigt) haben, einen Change Request implementiert haben oder neue Funktionalität hinzugefügt haben, freuen sich die OpenVAS-Entwickler über die Einsendung Ihrer Änderungen in Form eines Patches.

Um die Einbindung Ihres Patches in den Quelltext zu erleichtern sollten Sie die folgenden Richtlinien berücksichtigen:

- Falls möglich, erstellen Sie einen Patch gegen die aktuelle SVN-Revision.
- Schicken Sie Ihren Patch als einen „Unified Context Diff“, in dem Format, das das GNU-Programm `diff` erstellt, wenn es mit dem Parameter `-u3` aufgerufen wird. Falls Sie Ihre Arbeitskopie aus der SVN-Quelltextverwaltung ausgecheckt haben, wird der Aufruf des Befehls `svn diff` einen korrekten Patch erstellen.
- Bitte beschreiben Sie Ihre Änderungen auf englisch in dem jeweiligen ChangeLog. Dies hilft (besonders anderen Entwicklern) dabei, Ihre Änderungen und Ihre Gründe zu verstehen.
- Versuchen Sie Ihre Patches möglichst atomar zu halten. Das bedeutet, dass jeder Patch nur jeweils eine neue Funktionalität oder Fehlerbehebung beinhalten sollte. Bitte schicken Sie keinen Patch, der dutzende unzusammenhängende Änderungen enthält; es ist sehr unwahrscheinlich, dass die OpenVAS-Entwickler alle Ihre Änderungen gleichzeitig durchführen wollen. Ihr Patch hat eine sehr viel höhere Chance angenommen zu werden, wenn Sie eine Anzahl kleinerer Patches einschicken und es den Entwicklern überlassen, den optimalen Zeitpunkt für die Einbindung Ihrer Änderungen auszuwählen.

Wenn Sie diese Richtlinien befolgt haben, sollte Ihr Patch jetzt „versandfertig“ sein. Der beste Weg hierfür ist, den Patch an die Mailingliste der OpenVAS-Entwickler zu senden, die Sie unter

`openvas-devel@wald.intevation.org`

erreichen. Bitte erklären Sie in Ihrer Mail, was Sie geändert haben und warum Sie es geändert haben.

10.6 Schreibzugriff auf die Quelltextverwaltung

Schreibzugriff auf die Quelltextverwaltung wird von den Projekt-Koordinatoren erteilt. Wenn Sie Ihre Änderungen direkt in der Quelltextverwaltung vornehmen („committen“) wollen, fragen Sie einfach auf der Mailingliste der OpenVAS-Entwickler nach. In der Regel werden Sie dazu aufgefordert, Ihre ersten Änderungen als Patches einzuschicken um zu zeigen, dass Sie wissen, was Sie tun. Wenn diese Patches akzeptiert werden, erhalten Sie in der Regel sofort Schreibzugriff auf die Quelltextverwaltung.

Bitte beachten Sie, dass die Quelltextverwaltung von mehreren Entwicklern beobachtet wird. Änderungen von unzureichender Qualität und große unkoordinierte Änderungen können dazu führen, dass Ihre Änderungen von anderen Entwicklern wieder rückgängig gemacht („reverted“) werden.

10.7 ChangeLog führen

In jedem Hauptmodul von OpenVAS wird im Wurzelverzeichnis ein ChangeLog geführt. In diesem ChangeLog werden alle Änderungen im „ChangeLog“-Format aufgezeichnet. Das ChangeLog ist im GNU-Stil verfasst, die Syntaxhervorhebung sollte im Editor Ihrer Wahl im Allgemeinen automatisch funktionieren.

Einige Richtlinien für die ChangeLog-Datei:

- Behalten Sie den Syntax im GNU-Stil bei. Die Syntaxhervorhebung Ihres Editors sollte Sie dabei unterstützen.
- Führen Sie atomare Commits durch; committen Sie immer ein aktualisiertes ChangeLog mit Ihren Änderungen.
- Führen Sie jede geänderte Datei explizit im ChangeLog auf, auch wenn es sich um viele Dateien handelt. Sie können sich beim Schreiben Ihrer ChangeLog-Einträge an bereits vorhandenen Einträgen orientieren.

Beachten Sie: Das ChangeLog wird absichtlich nicht automatisch geführt. Dadurch werden Entwickler vor einem Commit dazu gezwungen, Ihre Änderungen nochmals zu überdenken. Dies führt regelmäßig dazu, dass suboptimale Änderungen oder Änderungen, die nicht in die Quelltextverwaltung gehören, vor dem Commit bemerkt werden.

Das ChangeLog der Entwickler dient als Basis für die dem Anwender zugängliche CHANGES-Datei bei der Vorbereitung einer neuen Veröffentlichung.

10.8 Quelltext Styleguide

Da verschiedene Formattierungs- Stile im OpenVAS Quelltext verwendet wurden (hauptsächlich in den von Nessus übernommenen Teilen), suchten die OpenVAS- Entwickler eine Übereinstimmung über Formatierungsregeln für Quelltext und Dokumentation.

Das Resultat dieses Changes Requestes war, dass den Empfehlungen des GNU Projektes

(http://www.gnu.org/prep/standards/standards.html#Formatting) zu folgen sei und man doxygen mit in-code Javadoc- Stil Kommentaren benutzen solle

(http://www.stack.nl/ dimitri/doxygen/docblocks.html).

Dadurch ist definiert wie neuer oder geänderter OpenVAS Quelltext idealerweise formatiert und dokumentiert sein sollte.

Eine Zusammenfassung der wichtigsten Regeln:

- Dateien: Pro Zeile maximal 80 Zeichen.
- Einrückung: Benutzen Sie reguläre Leerzeichen an Stelle von Tabulatoren.
- Einrückung: Ein Einrückungs- Schritt besteht aus zwei Leerzeichen.
- Definition von Funktionen: Benutzen Sie nicht den alten K&R Stil.
- Definition von Funktionen: Fügen Sie vor einer sich öffnenden Klammer ein Leerzeichen ein.
- Definition von Funktionen: Geben Sie dem Rückgabetyp eine eigene Zeile.
- Dokumentation von Funktionen: Dokumentieren Sie jede Funktion.

Es folgt eine beispielhafte Datei mit einigen englischen Kommentaren. Diese Datei ist in der Quelltextverwaltung nicht enthalten und sollte nicht kompiliert werden.

```
/* OpenVAS-Client
 * $Id$
 * Description: Proposed formating example (altered slad_install.c)
 *
 * Authors:
 * Felix Wolfsteller <felix.wolfsteller@intevation.de>
 *
 * Copyright:
 * Copyright (C) 2008 by Intevation GmbH
 *
 * 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, at your option, 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.
 *
 * In addition, as a special exception, you have
 * permission to link the code of this program with the OpenSSL
 * library (or with modified versions of OpenSSL that use the same
 * license as OpenSSL), and distribute linked combinations including
 * the two. You must obey the GNU General Public License in all
 * respects for all of the code used other than OpenSSL. If you
 * modify this file, you may extend this exception to your version
 * of the file, but you are not obligated to do so. If you do not
 * wish to do so, delete this exception statement from your version.
 */


#include <unistd.h>
#include <gtk/gtk.h>
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "nessus_i18n.h"
#include "module.h"


/**
 * Defines follow the includes which follow the header, are capitalized and can
 * be documented.\n
 * Documentation is done in the source file, not the header file.
 */
#define SOMEDEFINE "fine"
```

```
/**
 * Static functions do not neccessarily begin with the module name.
 * @param str Parameters can be described like this.
 */
static void
statfunc (gchar *str)
{
  /* Indentation by 2 blanks, single lines should not exceed 80 chars */
  GtkWidget* m = gtk_message_dialog_new (GTK_WINDOW(NULL),
                                         (GTK_DIALOG_MODAL
                                          | GTK_DIALOG_DESTROY_WITH_PARENT),
                                         GTK_MESSAGE_ERROR, GTK_BUTTONS_OK,
                                         str);
  /* (If 80 chars per line do not suffice, try to break it up nicely) */
  gtk_dialog_run (GTK_DIALOG (m));
  gtk_object_destroy (GTK_OBJECT (m));
}


/**
 * @brief For an overview, I can describe what happens here in one sentence.
 *
 * Non-static functions begin with the name of the module + underscore.
 * @param win Widget to work on.
 * @param data Data to work with.
 * @return String that has to be freed.
 */
char*
module_func (GtkWidget* win, gpointer data)
{
  char buf[1024];
  pid_t p;
  FILE* f;
  /* Structuring by newlines is fine. */

  f = popen (SLADINSTALLER " --ping", "r");

  /* Conditional and loop intendation same as for functions. */
  if (!f)
    {
      statfunc (_("Could not check SLAD installer."));
      return;
    }
  else /* f != NULL */
    {
      statfunc (_("An else"));
    }
  fgets (buf, sizeof (buf), f);
  pclose (f);

#ifndef CYGWIN
  /* Compiled with flag -mwindows, sladinstaller
```

```
   * does not answer on stdout.
   * Therefore, this test is only performed on
   * non-CYGWIN systems and simply dropped for
   * CYGWIN. */
  if (strcmp (buf, "pong\n"))
    {
      statfunc (_("Could not execute SLAD installer."));
      return;
    }
#endif /* CYGWIN */

  // Single-line intermediate comments can also start by double slashes.
  p = fork ();
}
```

11 OpenVAS Transfer Protocol (OTP)

In einer OpenVAS-Installation kommunizieren das Client- und Server-Modul mittels des OpenVAS Transfer Protocols (OTP). Frühere OpenVAS-Versionen benutzten das von Nessus übernommene Nessus Transport Protocol (NTP). Um Mängel im NTP auszugleichen und weitere Verbesserungen einführen zu können, wurde es notwendig, Änderungen am Protokoll einzubringen.

Da das NTP vom Nessus-Projekt spezifiziert wurde und Änderungen am Protokoll durch das Nessus-Projekt zu erwarten sind, wurde entschieden, ein anderes Protokoll zu verwenden. Damit sollen Problemen mit zukünftigen Spezifikationen durch das Nessus-Projekt und Verwechselungen mit anderen etablierten Protokollen entgegen gewirkt werden.

Die erste OTP Spezifikation ist der letzten unter der GNU Public Licence (GPL) veröffentlichten Version des NTP sehr ähnlich.

11.1 Änderungen von NTP 1.2 zu OTP 1.0

Dieser Abschnitt beschreibt die Unterschiede zwischen NTP 1.2 und OTP 1.0. Sollten Sie bereits Erfahrungen mit dem NTP haben, wird er Ihnen die Hauptunterschiede zwischen den beiden Protokollen aufzeigen.

Hochladen von NVTs Abschnitt 10 der „NTP Extensions“ beschreibt den Nachrichtentyp `ATTACHED_PLUGIN`. Dieser Nachrichtentyp machte es einer Client-Anwendung möglich, ein NVT auf einen Server hochzuladen. Aufgrund von Sicherheitsbedenken, die im OpenVAS Change Request #4 beschrieben sind, wurde dieser Nachrichtentyp vom Protokoll entfernt.

Versionsinformationen Der undokumentierte Nachrichtentyp `NESSUS_VERSION` wurde durch den `OPENVAS_VERSION` Nachrichtentyp ersetzt. Auf eine vom Client gesendete `OPENVAS_VERSION` Nachricht wird als Antwort eine Nachricht mit der aktuellen Versionsnummer des Servers erwartet.

Neue Nachrichtentypen Zu den bestehenden Nachrichtentypen `HOLE`, `INFO` und `NOTE` wurden zwei weitere Typen, `DEBUG` und `LOG`, hinzugefügt. Sie erlauben dem Nutzer, sich auch Debug- und Log-Nachrichten der NVTs anzusehen.

Detached Scans Die Funktionalität, den Server Scans ausführen zu lassen, ohne dass eine Verbindung zu einem Client besteht, wurde aufgrund von Designentscheidungen aufgegeben. In der Folge wurden die Kommandos `DETACHED_SESSIONS_LIST` und `DETACHED_STOP` vom Protokoll entfernt. In der weiteren Konsequenz wurden auch folgende damit zusammenhängende Voreinstellungen entfernt: detached_scan, continuous_scan, delay_between_scan_loops, detached_scan_email_address.

Informationen über die Reihenfolge der NVTs Das Server-Kommando `PLUGINS_ORDER` wurde im NTP 1.2 definiert aber nicht implementiert. Es wurde vom Protokoll entfernt.

Scans anstoßen Im NTP gab es zwei Möglichkeiten, einen Scan anzustoßen, `NEW_ATTACK` und `LONG_ATTACK`. Letzteres ermöglichte beliebig lange Listen von Zielen, während es für `NEW_ATTACK` eine Begrenzung von 4000 Bytes gab. Da sowohl der Nessus- als auch der OpenVAS-Client nur `LONG_ATTACK` benutzen, wurde `NEW_ATTACK` aus dem Protokoll entfernt.

Benachrichtung bei fehlerhaften Einstellungen Das NTP erlaubte die Benachrichtigung des Clients im Falle fehlerhafter Einstellungen. Dies sollte mit dem Nachrichtentyp `PREFERENCES_ERRORS` erreicht werden. Aus Designgründen und da der Nachrichtentyp weder im Server noch im Client je vollständig implementiert worden war, wurde er aus dem Protokoll gestrichen.

Protokoll-Erweiterungen Die folgenden Protokoll-Erweiterungen wurden zum Standard des OTP Protokolls erhoben: „timestamps“, „dependencies“, „plugins_version“, „plugins_cve_id“, „plugins_bugtraq_id“ und „plugins_xrefs“.

11.2 Allgemeines zum OTP

Das OpenVAS Transfer Protocol ist textbasiert, für Menschen lesbar und zeilenorientiert. Jede Zeile besteht aus Feldern, die durch „ `<|>` “ getrennt werden. Das jeweils erste Feld lässt erkennen, ob es sich um eine vom Server oder vom Client stammende Nachricht handelt („`CLIENT`“ vs. „`SERVER`“).

11.3 Protokoll Initialisierung

Eine Sitzung wird vom Client initiiert, indem er einen Versions-String sendet, der die entsprechende Protokollversion vom Server anfragt. Auf diese Nachricht kann der Server mit dem selben Versions-String antworten wenn er die angefragte Protokollversion unterstützt oder die Verbindung zum Client beenden.

Syntax:

```
< OTP/1.0 >
User : nutzer_name
Password : nutzer_passwort
```

11.4 Protokoll-Kommandos

11.4.1 ATTACHED_FILE

Beschreibung: Dieses Kommando entspricht dem NVT Einstellungstyp „Datei“. Es folgt auf das Kommando `PREFERENCES`, um eine Datei vom Client auf den Server hochzuladen.

Syntax:

```
CLIENT <|> ATTACHED_FILE
name: datei_name
content: octet/stream
bytes: datei_laenge
datei_inhalt
```

wobei

datei_name für den Dateinamen steht (exakt dieser Name wird in den NVT Einstellungen referenziert).

datei_laenge die Anzahl der Bytes, die nach dem Zeilenumbruch folgen.

datei_inhalt die tatsächliche Datei als Bytestream ist.

11.4.2 BYE

Beschreibung: Dieses Kommando wird vom Server benutzt, um das Ende einer Sitzung mitzuteilen.

Der Client sollte dieses Kommando wie aufgezeigt bestätigen (ACK für engl. „acknowledge“).

Syntax:

```
SERVER <|> BYE <|> BYE <|> SERVER
CLIENT <|> BYE <|> ACK
```

11.4.3 CERTIFICATES

Beschreibung: Dieses Kommando wird vom Client benutzt um Zertifikate anzufragen. Der Server verwendet es, um eben diese zu versenden. Enthalten sind der Fingerabdruck, der Name des Zertifikatbesitzers, der Vertrauensgrad, und der öffentliche Schlüssel selbst.

Syntax:

```
CLIENT <|> CERTIFICATES <|> CLIENT
SERVER <|> CERTIFICATES
fingerabdruck <|> besitzer_name <|> vertrauensgrad <|> laenge_in_bytes <|>
pubkey
<|> SERVER
```

wobei

fingerabdruck ein 48 Bytes großes Feld ist, in dem der Fingerabdruck des Zertifikates steht.

besitzer_name den Besitzernamen angibt.

vertrauensgrad entweder „trusted“ (vertraut) oder „notrust“ (kein vertrauen) ist.

laenge_in_bytes die Anzahl kommender bytes angibt (Schlüssellänge).

pubkey der ASCII-enkodierte öffentliche Schlüssel selber ist, wobei Zeilenumbuchzeichen durch Semikola ersetzt wurden.

11.4.4 COMPLETE_LIST

Beschreibung: Dieses Kommando kann vom Client benutzt werden um eine Liste aller NVTs vom Server anzufragen.

Normalerweise folgt es den `PLUGINS_MD5` Kommandos des Servers, wenn die Prüfsumme (md5sum) des Clientseitigen NVT-Zwischenspeichers nicht mit der des Servers übereinstimmt.

Alternativ dazu kann der Client auch das Kommando `SEND_PLUGINS_MD5` verwenden.

Der Server wird mit dem Kommando `PLUGIN_LIST` antworten.

Syntax:

```
CLIENT <|> COMPLETE_LIST <|> CLIENT
```

11.4.5 DEBUG

Beschreibung: Mit diesem Kommando meldet der Server ein identifiziertes Problem der Klasse „debug“. Wenn zu der Meldung kein Port in Verbindung steht, wird an der entsprechenden Stelle der Nachricht „general“ stehen.

Syntax:

```
SERVER <|> DEBUG <|> host <|> service_name (port_nummer/protokoll_typ) <|>
beschreibung <|> oid <|> SERVER

SERVER <|> DEBUG <|> host <|> general <|> beschreibung <|> oid <|> SERVER
```

wobei

host das Zielsystem angibt.

service_name für den Name des Dienstes (wie in /etc/services) steht.

port_nummer die Nummer des Ports, auf das sich das Problem bezieht, ist.

protokoll_typ entweder „tcp“ oder „udp“ ist.

beschreibung das Problem beschreibt. Zeilenumbrüche wurden durch Semikola ersetzt.

oid die OID des NVTs ist, der das Problem identifiziert hat.

11.4.6 ERROR

Beschreibung: Sollten Probleme auftreten wird der Server mit diesem Kommando eine Fehlermeldung verschicken.

Sollten die Probleme nicht behebbar sein, wird der Server danach die Verbindung zum Client mit dem `BYE` Kommando beenden.

Syntax:

```
SERVER <|> ERROR <|> beschreibung des fehlers <|> SERVER
```

11.4.7 FINISHED

Beschreibung: Dieses Kommando wird jedes Mal vom Server versendet, wenn der Scan eines Zielrechners abgeschlossen ist.

Dieses Verhalten des Servers lässt sich vom Client mit der Einstellung „ntp_opt_show_end“ an- oder abschalten.

Syntax:

```
SERVER <|> FINISHED <|> host <|> SERVER
```

11.4.8 GO ON

Beschreibung: Dieses Kommando kann vom Client verwendet werden um zu bestätigen, dass die Informationen über die NVTs erhalten wurden und signalisiert dem Server, dass er fortfahren kann.

Normalerweise folgt es den `PLUGINS_MD5` Kommandos des Servers, wenn die Prüfsumme (md5sum) des clientseitigen NVT-Zwischenspeichers mit der des Servers übereinstimmte.

Der Server wird darauf mit dem Kommando `PREFERENCES` antworten und die Kommunikation fortsetzen.

Syntax:

```
CLIENT <|> GO ON <|> CLIENT
```

11.4.9 HOLE

Beschreibung: Mit diesem Kommando meldet der Server ein identifiziertes Problem der Klasse „Sicherheitslücke“. Wenn zu der Sicherheitslücke kein Port in Verbindung steht, wird an der entsprechenden Stelle der Nachricht „general“ stehen.

Syntax:

```
SERVER <|> HOLE <|> host <|> service_name (port_nummer/protokoll_typ) <|>
beschreibung <|> oid <|> SERVER

SERVER <|> HOLE <|> host <|> general <|> beschreibung <|> oid <|> SERVER
```

wobei

host das Zielsystem angibt.

service_name für den Name des Dienstes (wie in /etc/services) steht.

port_nummer die Nummer des Ports, auf das sich das Problem bezieht, ist.

protokoll_typ entweder „tcp“ oder „udp“ ist.

beschreibung das Problem beschreibt. Zeilenumbrüche wurden durch Semikola ersetzt.

oid die OID des NVTs ist, der das Problem identifiziert hat.

11.4.10 INFO

Beschreibung: Mit diesem Kommando meldet der Server ein identifiziertes Problem der Klasse „Sicherheitsinformation“. Wenn zu der Meldung kein Port in Verbindung steht, wird an der entsprechenden Stelle der Nachricht „general“ stehen.

Syntax:

```
SERVER <|> INFO <|> host <|> service_name (port_nummer/protokoll_typ) <|>
beschreibung <|> oid <|> SERVER

SERVER <|> INFO <|> host <|> general <|> beschreibung <|> oid <|> SERVER
```

wobei

host das Zielsystem angibt.

service_name für den Name des Dienstes (wie in /etc/services) steht.

port_nummer die Nummer des Ports, auf das sich das Problem bezieht, ist.

protokoll_typ entweder „tcp“ oder „udp“ ist.

beschreibung das Problem beschreibt. Zeilenumbrüche wurden durch Semikola ersetzt.

oid die OID des NVTs ist, der das Problem identifiziert hat.

11.4.11 LOG

Beschreibung: Mit diesem Kommando meldet der Server ein identifiziertes Problem der Klasse „log“. Wenn zu der Meldung kein Port in Verbindung steht, wird an der entsprechenden Stelle der Nachricht „general“ stehen.

Syntax:

```
SERVER <|> LOG <|> host <|> service_name (port_nummer/protokoll_typ) <|>
beschreibung <|> oid <|> SERVER

SERVER <|> LOG <|> host <|> general <|> beschreibung <|> oid <|> SERVER
```

wobei

host das Zielsystem angibt.

service_name für den Name des Dienstes (wie in /etc/services) steht.

port_nummer die Nummer des Ports, auf das sich das Problem bezieht, ist.

protokoll_typ entweder „tcp“ oder „udp“ ist.

beschreibung das Problem beschreibt. Zeilenumbrüche wurden durch Semikola ersetzt.

oid die OID des NVTs ist, der das Problem identifiziert hat.

11.4.12 LONG_ATTACK

Beschreibung: Mit diesem Kommando fordert der Client den Server auf, ein oder mehrere Zielsystem(e) anzugreifen. Das oder die Zielsystem(e) sind in „hosts“ durch Komma getrennte IPs oder FQDNs anzugeben.

Der Parameter „laenge“ gibt die Anzahl der Bytes von „hosts“ an. Sollten „laenge“ und die tatsächliche Anzahl der Bytes in „hosts“ nicht übereinstimmen, beendet der Server die Verbindung zum Client.

Die Kommandos `PREFERENCES` und `RULES` sollten dem Kommando `LONG_ATTACK` vorausgehen.

Syntax:

```
CLIENT <|> LONG_ATTACK
laenge
hosts
```

11.4.13 NOTE

Beschreibung: Mit diesem Kommando meldet der Server ein identifiziertes Problem der Klasse „Sicherheitshinweis“. Wenn zu dem Hinweis kein Port in Verbindung steht, wird an der entsprechenden Stelle der Nachricht „general“ stehen.

Syntax:

```
SERVER <|> NOTE <|> host <|> service_name (port_nummer/protokoll_typ) <|>
beschreibung <|> oid <|> SERVER

SERVER <|> NOTE <|> host <|> general <|> beschreibung <|> oid <|> SERVER
```

wobei

host das Zielsystem angibt.

service_name für den Name des Dienstes (wie in /etc/services) steht.

port_nummer die Nummer des Ports, auf das sich das Problem bezieht, ist.

protokoll_typ entweder „tcp“ oder „udp“ ist.

beschreibung das Problem beschreibt. Zeilenumbrüche wurden durch Semikola ersetzt.

oid die OID des NVTs ist, der das Problem identifiziert hat.

11.4.14 OPENVAS_VERSION

Beschreibung: Mit diesem Kommando fragt der Client die Version des Servers an.

Der Server wird wie folgt antworten:

Syntax:

```
CLIENT <|> OPENVAS_VERSION <|> CLIENT

SERVER <|> OPENVAS_VERSION <|> version <|> SERVER
```

11.4.15 PLUGINS_DEPENDENCIES

Beschreibung: Die `PLUGINS_DEPENDENCIES` Nachricht wird nach der `RULES` Nachricht verschickt.

Syntax:

```
SERVER <|> PLUGINS_DEPENDENCIES
nvt_name1 <|> dependency1 <|> dependency2 <|> ... <|>
nvt_name2 <|> dependency1 <|> dependency2 <|> ... <|>
...
<|> SERVER
```

11.4.16 PLUGINS_MD5

Beschreibung: Achtung: Dieses Kommando tritt in zwei Fällen auf.

1. Es kann an Stelle des `PLUGIN_LIST` Kommandos verwendet werden. „md5sum“ beschreibt dann die Prüfsumme über alle NVTs.
2. Es kann einem `SEND_PLUGINS_MD5` Kommando folgen. „md5sum“ ist dann die Prüfsumme jedes einzelnen NVT.

Syntax:

1. `SERVER <|> PLUGINS_MD5 <|> md5sum <|> SERVER`
2. ```
   SERVER <|> PLUGINS_MD5
   nvt_oid1 <|> md5sum1
   nvt_oid2 <|> md5sum2
   ...
   <|> SERVER
   ```

11.4.17 PLUGIN_INFO

Beschreibung: Dieses Kommando wird vom Client benutzt um Informationen über einen NVT anzufragen. Der NVT wird dabei über seine OID identifiziert.

Syntax:

```
CLIENT <|> PLUGIN_INFO <|> oid <|> CLIENT
```

Der Server antwortet (analog zu dem `PLUGIN_LIST` Kommando) mit:

```
oid <|> name <|> kategorie <|> copyright <|> beschreibung <|> zusammenfassung
<|> familie  <|> plugin_version <|> cve_id <|> bugtraq_id <|> xrefs <|> fprs
```

Das letzte Feld (fprs) ist eine Komma- separierte Liste von Fingerabdrücken der Signaturen, falls vorhanden.

Anstelle der cve_id, bugtraq_id, xrefs und fprs werden symbolische Werte (NOCVE, NOBID, NOXREFS, NOSIGNKEYS) geschickt, falls keine cve_id, bugtraq_id usw. gefunden werden kann.

Falls kein NVT mit der OID=oid gefunden werde sollte, wird der Server nicht antworten.
```

11.4.18 PLUGIN_LIST

Beschreibung: Mit diesem Kommando schickt der Server detaillierte Informationen über die verfügbaren NVTs.

Direkt nach diesem Kommando werden vom Server die `PREFERENCES` und `RULES` Kommandos gesendet.

Der Client kann Informationen zu einzelnen NVTs mit dem Kommando `PLUGIN_INFO` anfordern.

Syntax:

```
SERVER <|> PLUGIN_LIST <|>
oid <|> name <|> kategorie <|> copyright <|> beschreibung <|> zusammenfassung
<|> familie <|> plugin_version <|> cve_id <|> bugtraq_id <|> xrefs <|> fprs
oid <|> name <|> kategorie <|> copyright <|> beschreibung <|> zusammenfassung
<|> familie <|> plugin_version <|> cve_id <|> bugtraq_id <|> xrefs <|> fprs
...
<|> SERVER
```

Das letzte Feld (fprs) ist eine Komma- separierte Liste von Fingerabdrücken der Signaturen, falls vorhanden.

Anstelle der cve_id, bugtraq_id, xrefs und fprs werden symbolische Werte (NOCVE, NOBID, NOXREFS, NOSIGNKEYS) geschickt, falls keine cve_id, bugtraq_id usw. gefunden werden kann.

11.4.19 PORT

Beschreibung: Mit diesem Kommando meldet der Server einen offenen Port auf dem Zielsystem „host“.

Syntax:

```
SERVER <|> PORT <|> host <|> port_nummer <|> SERVER
```

11.4.20 PREFERENCES

Beschreibung: Mit diesem Kommando werden Werte für die Einstellungen ausgetauscht. Der Server benutzt es, um den Client über die Voreinstellungen zu informieren, der Client benutzt es dann um die Wahl des Benutzers zu übermitteln.

Hinweis: Die Syntaxdefinition beschreibt neben dem Anwenden allgemeiner Einstellungen auch die gesonderte Art und Weise, wie Einstellung einzelner NVTs vorgenommen werden können.

Vorhandene Einstellungen:

ntp_save_sessions Lässt serverseitiges Speichern von Scansitzungen zu, wenn auf „yes“ gesetzt. Dadurch werden folgende Kommandos verfügbar:

- `SESSIONS_LIST`
- `SESSION_DELETE`
- `SESSION_RESTORE`

save_session Der Server wird den Scan als Sitzung speichern, wenn auf „yes“ gesetzt.

save_empty_sessions Wird nur beachtet, wenn save_session auf „yes“ gesetzt wurde. Dann würden auch leere Scans als Sitzung gespeichert.

max_threads

test_file

ping_hosts

reverse_lookup

outside_firewall

host_expansion

port_range

max_hosts

save_knowledge_base Aktiviert das Speichern der Wissensbasis, wenn „yes“.

only_test_hosts_whose_kb_we_have Wenn auf „yes“ gesetzt, wird ein Scan nur für Zielsysteme ausgeführt, für die die Wissensbasis gefüllt ist.

only_test_hosts_whose_kb_we_dont_have Wenn auf „yes“ gesetzt, wird ein Scan nur für Zielsysteme ausgeführt, für die die Wissensbasis leer ist.

kb_restore Stellt den Inhalt der Wissensbasis für getestete Zielsysteme wieder her, wenn auf „yes“ gesetzt.

kb_dont_replay_scanners Falls diese Einstellung ebenso wie kb_restore auf „yes“ gesetzt sind und Einträge in der Wissensbasis vorhanden sind werden die Scanner nicht ausgeführt.

kb_dont_replay_info_gathering Falls diese Einstellung ebenso wie kb_restore auf „yes“ gesetzt sind und Einträge in der Wissensbasis vorhanden sind werden die „Informations Sammler“ NVTs nicht ausgeführt.

kb_dont_replay_attacks Falls diese Einstellung ebenso wie kb_restore auf „yes“ gesetzt sind und Einträge in der Wissensbasis vorhanden sind werden Angriffs-Skripte nicht ausgeführt.

kb_dont_replay_denials Falls diese Einstellung ebenso wie kb_restore auf „yes“ gesetzt sind und Einträge in der Wissensbasis vorhanden sind werden die DoS-NVTs nicht ausgeführt.

kb_max_age Diese Einstellung setzt Alter (in Sekunden) einer Wissensbasis, ab dem sie als veraltet gilt.

timeout.<nvt_id> = <timeout> Setzt die Laufzeitbegrenzung (timeout) für den NVT <nvt_id>. Der Wert „-1“ wird als keine spezifische Laufzeitbegrenzung interpretiert.

Nur vom Client verwendet:

plugin_set wenn leer, bedeutet dies „alle NVTs“.

ntp_opt_show_end lässt den Server `FINISHED` Nachrichten schicken.

ntp_keep_communication_alive Lässt den Server die Verbindung halten, nachdem ein Scan abgeschlossen wurde.

ntp_short_status Lässt den Server verkürzte `STATUS` Nachrichten verschicken, um die Datenübertragung zu verringern.

Syntax:

```
SERVER <|> PREFERENCES <|>
einstellung_name <|> wert
einstellung_name <|> wert
einstellung_name <|> wert
...
<|> SERVER

CLIENT <|> PREFERENCES <|>
einstellung_name <|> wert
einstellung_name <|> wert
einstellung_name <|> wert
...
<|> CLIENT
```

Für Einstellung individueller NVTs können Zeilen dieser Art innerhalb der Liste verwendet werden:

```
nvt_name[einstellunhgs_typ]:einstellung_name <|> wert
```

wobei

nvt_name den NVT spezifiziert, für den die Einstellung gesetzt wird.

einstellung_typ den Wertetyp bestimmt, der schließlich verwendet wird um zu entscheiden, wie die Auswählmöglichkeit in der Benutzungsoberfläche aussehen wird.

einstellung_name die Einstellung bestimmt (wird in der Benutzungsoberfläche angezeigt).

wert die Voreinstellung ist, wenn die Nachricht vom Server geschickt wurde. Wenn vom Client geschickt, entspricht „wert“ dem entsprechend ausgewähltem Wert.

Folgende Werte können von einstellung_typ angenommen werden:

checkbox Wert ist „yes“ oder „no“.

entry Wert ist eine Zeichenkette.

password Wert ist ebenfalls ein Zeichenkette, soll aber in der grafischen Oberfläche und in Textdateien nicht als solcher angezeigt werden.

radio Wert ist eine Liste von Komma-separierten Optionen, wenn vom Server stammend, oder nur die entsprechend ausgewählte Option wenn vom Client stammend.

file Wert ist „<none>“ wenn vom Server gesendet und ein Dateipfad wenn vom Client gesendet. Der Client muss die Datei mit der gleichen Pfadangabe mit dem Kommando `ATTACHED_FILE` übermitteln.

11.4.21 RULES

Beschreibung: Regeln (zu engl. „rules“) schränken die mögliche Auswahl von Zielsystemen ein. Vom Client definierte Regeln 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
```

11.4.22 SEND_PLUGINS_MD5

Beschreibung: Dieses (Kommando) folgt normalerweise auf ein `PLUGINS_MD5` Kommando des Servers, wenn die serverseitige Prüfsumme nicht mit der Prüfsumme über die auf dem Client zwischengespeicherten NVTs übereinstimmt.

Der Client kann alternativ das Kommando `COMPLETE_LIST` benutzen.

Der Server wird mit dem Kommando `PLUGINS_MD5` antworten.

Syntax:

```
CLIENT <|> SEND_PLUGINS_MD5 <|> CLIENT
```

11.4.23 SESSIONS_LIST

Beschreibung: Mit diesem Kommando fragt der Client die Liste von Sitzungen für den angemeldeten Nutzer an. Die Sitzungen liegen gespeichert auf dem Server vor.

Der Server wird mit dem selben Kommando antworten und die Liste der Sitzungen übertragen. Die Namen der Sitzungen werden von den Zeitstempel abgeleitet. Die Hosts sind die Ziele, die für die jeweilige Sitzung angegeben wurden. Die Hosts werden nur angegeben um die Sitzungen identifizieren zu können. Die Liste wird nach 4000 Bytes abgeschnitten.

Syntax:

```
CLIENT <|> SESSIONS_LIST <|> CLIENT

SERVER <|> SESSIONS_LIST
sitzungs_name1 hosts1
sitzungs_name2 hosts2
...
<|> SERVER
```

11.4.24 SESSION_DELETE

Beschreibung: Mit diesem Kommando entfernt der Client eine durch ihren Namen identifizierte Sitzung aus dem serverseitigem Speicher. Der Server antwortet nur im Falle eines Fehlers mit dem `ERROR` Kommando und bleibt im Erfolgsfalle still.

Syntax:

```
CLIENT <|> SESSION_DELETE <|> sitzungs_name <|> CLIENT
```

11.4.25 SESSION_RESTORE

Beschreibung: Mit diesem Kommando fordert der Client den Server auf, eine durch „sitzungs_name“ zu identifizierende Sitzung wieder aufzunehmen. Der Server wird sich verhalten als ob er ein `LONG_ATTACK` Kommando erhalten hätte und sofort alle bisher in dieser Sitzung gesammelten Resultate zurücksenden. Daraufhin wird er den Scan dort fortsetzen, wo er unterbrochen wurde.

Syntax:

```
CLIENT <|> SESSION_RESTORE <|> sitzungs_name <|> CLIENT
```

11.4.26 STATUS

Beschreibung: Mit diesem Kommando wird der Client über den Fortschritt des Scans des Zielsystems „host“ informiert. In dem „attack_state“-Feld steht entweder „portscan“ oder „attack“ (als „p“ und „a“ abgekürzt, wenn der Client die Option „ntp_short_status“ angewählt hat). Das fünfte Feld („aktuell/max“) zeigt an, welcher Port momentan angesprochen wird und welches der letzte („maximale“ - mit der höchsten Portnummer) Port sein wird.

Syntax:

```
SERVER <|> STATUS <|> host <|> attack_state <|> current/max <|> SERVER
```

Falls der Client die Einstellung „ntp_short_status“ angeschaltet hat:

```
SERVER <|> STATUS <|> attack_state:host:current:max <|> SERVER
```

11.4.27 STOP_ATTACK

Beschreibung: Mit diesem Kommando fordert der Client den Server auf, den Scan des Zielsystems „host“ abzubrechen.

Syntax:

```
CLIENT <|> STOP_ATTACK <|> host <|> CLIENT
```

11.4.28 STOP_WHOLE_TEST

Beschreibung: Dieses Kommando wird vom Client geschickt, wenn der gesamte momentan laufende Test abgebrochen werden soll.

Syntax:

```
CLIENT <|> STOP_WHOLE_TEST <|> CLIENT
```

11.4.29 TIME

Beschreibung: Die `TIME` Nachricht wird vom Server gesendet um über die Dauer des Scans eines einzelnen Hosts, sowie über die Dauer des gesamten Scans zu informieren.

Syntax: Nachdem der Scan eines Zielsystems abgeschlossen wurde sendet der Server:

```
SERVER <|> TIME <|> HOST_START <|> host <|> zeit_string <|> SERVER
SERVER <|> TIME <|> HOST_END <|> host <|> zeit_string <|> SERVER
```

oder, falls ein `STOP\_ATTACK` Kommando vom Client geschickt wurde:

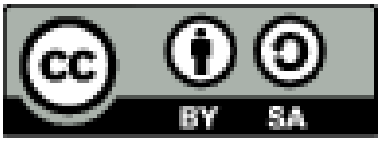
```
SERVER <|> TIME <|> HOST_START <|> host <|> zeit_string <|> SERVER
SERVER <|> TIME <|> HOST_INTERRUPTED <|> host <|> zeit_string <|> SERVER
```

Nachdem der gesamte Scan durchgeführt wurde:

```
SERVER <|> TIME <|> SCAN_START <|> zeit_string <|> SERVER
SERVER <|> TIME <|> SCAN_END <|> zeit_string  <|> SERVER
```

wobei zeit_string das Format „Wed Jun 30 21:49:08 1993“ annimmt.

12 Document License: CC by SA



Creative Commons Attribution-ShareAlike 3.0 Unported

You are free to copy, distribute and transmit the work under the following conditions:

Attribution. You must give the original author credit.

Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of the above conditions can be waived if you get permission from the copyright holder.

Nothing in this license impairs or restricts the author's moral rights.
To view the full licensing agreement, visit

http://creativecommons.org/licenses/by-sa/3.0/

or mail to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.