

OPSEC

Check Point™ VPN-1/FireWall-1® SCV (Secure Configuration Verification) Specification

NG Feature Pack 2

SCV (Secure Configuration Verification) Specification

November 2001

CHECK POINT™
Software Technologies Ltd.



Contents

Overview	7	Using an SCV DLL with VPN-1/FireWall-1 SecureClient	19
SCV Check	7	Integration with VPN-1/FireWall-1 SecureClient	19
Desktop	9	Installation	19
Programming Model	9	SCV Policy Example (local.scv)	20
Third Party Programming Model	9	SCV Policy Description	20
OPSEC programming model	9	SCV Check Tool	21
Aspects of Policy Server Downloads	9	Activating the Tool	21
SCV API Overview	10	Checktool Menu	21
SCV Test Tool	10	Life Cycle of SCV DLL	22
Overview	11	Sequence for Running Checktool	22
OPSEC Files	11	Passing Arguments to the Start Callback (argc, argv)	23
OPSEC Table of Libraries	12	Debugging SCV DLL	23
General API's	12	SCVEditor	23
UserMessageBox	12	Specific Commands in the SCVEditor	24
LogScv	12	SCV Product	25
NotifySCVStatus	13	SCV Category	26
UserAllocateString	13	Global SCV Parameters	27
ImpersonateUser	14		
RevertSelf	14		
IsUserLoggedOn	14		
Call Back Functions	15		
GetScvRegistrationParams	15		
Start	16		
Stop	16		
Init	16		
Clean	17		
Status	17		
GetScvDiagnostics	17		
Configuration	18		
How to create the DLL	18		

© 2000-2001 Check Point Software Technologies Ltd.

All rights reserved. This product and related documentation are protected by copyright and distributed under licensing restricting their use, copying, distribution, and decompilation. No part of this product or related documentation may be reproduced in any form or by any means without prior written authorization of Check Point. While every precaution has been taken in the preparation of this book, Check Point assumes no responsibility for errors or omissions. This publication and features described herein are subject to change without notice.

RESTRICTED RIGHTS LEGEND:

Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 and FAR 52.227-19.

TRADEMARKS:

Check Point, the Check Point logo, FireWall-1, FloodGate-1, INSPECT, IQ Engine, Open Security Extension, OPSEC, Provider-1, VPN-1 Accelerator Card, VPN-1 Certificate Manager, VPN-1 Gateway, VPN-1 Appliance, VPN-1 SecureRemote, ConnectControl, VPN-1 SecureServer and UserAuthority are trademarks or registered trademarks of Check Point Software Technologies Ltd. Meta IP and User-to-Address Mapping are trademarks of MetalInfo, Inc., a wholly-owned subsidiary of Check Point Software Technologies, Inc. RealSecure is a trademark of Internet Security Systems, Inc. All other product names mentioned herein are trademarks or registered trademarks of their respective owners.

The products described in this document are protected by U.S. Patent No. 5,606,668 and 5,835,726 and may be protected by other U.S. Patents, foreign patents, or pending applications.

THIRD PARTIES:

Entrust is a registered trademark of Entrust Technologies, Inc. in the United States and other countries. Entrust's logos and Entrust product and service names are also trademarks of Entrust Technologies, Inc. Entrust Technologies Limited is a wholly owned subsidiary of Entrust Technologies, Inc. FireWall-1 and SecuRemote incorporate certificate management technology from Entrust. Verisign is a trademark of Verisign Inc.

Copyright © 1996-1998. Internet Security Systems, Inc. All Rights Reserved. RealSecure, SAFEsuite, Intranet Scanner, Internet Scanner, Firewall Scanner, and Web Scanner are trademarks or registered trademarks of Internet Security Systems, Inc.

The following statements refer to those portions of the software copyrighted by University of Michigan.

Portions of the software copyright © 1992-1996 Regents of the University of Michigan. All rights reserved. Redistribution and use in source and binary forms are permitted provided that this notice is preserved and that due credit is given to the University of Michigan at Ann Arbor. The name of the University may not be used to endorse or promote products derived from this software without specific prior written permission. This software is provided "as is" without express or implied warranty.

Copyright © Sax Software (terminal emulation only).

The following statements refer to those portions of the software copyrighted by Carnegie Mellon University.

Copyright 1997 by Carnegie Mellon University. All Rights Reserved.

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of CMU not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

CMU DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL CMU BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

The following statements refer to those portions of the software copyrighted by The Open Group.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE OPEN GROUP BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

The following statements refer to those portions of the software copyrighted by The OpenSSL Project.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>).

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT "AS IS" AND ANY * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The following statements refer to those portions of the software copyrighted by Eric Young.

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Check Point Software Technologies Ltd.

International Headquarters:
3A Jabotinsky Street
Ramat Gan 52520, Israel
Tel: 972-3-753 4555
Fax: 972-3-575 9256
e-mail: info@CheckPoint.com

U.S. Headquarters:
Three Lagoon Drive, Suite 400
Redwood City, CA 94065
Tel: 800-429-4391 ; (650) 628-2000
Fax: (650) 654-4233
<http://www.checkpoint.com>

Please direct all comments regarding this publication to techwriters@checkpoint.com.



Preface

Scope

This document describes the Scv (Secure Configuration Verification) Specification.

Who Should Use this Document

This API specification is written for developers who write software to enhance the network security provided by VPN-1/Firewall-1.

It also assumes that you have a basic understanding and a working knowledge of the following:

- system and network security
- the VPN-1/FireWall-1 product
- system and network administration
- the C and/or C++ programming language
- the Unix or Windows operating system
- Internet protocols

What Typographic Variations Mean

The following table describes the typographic variations used in this book.

TABLE P-1 Typographic Conventions

Typeface or Symbol	Meaning	Example
AaBbCc123	The names of commands, files, and directories; on-screen computer output; code	Edit your .login file. Use <code>ls -a</code> to list all files. machine_name% You have mail. <code>session = sam_new_session (client, server);</code>
AaBbCc123	same as above, but with emphasis	session = sam_new_session (client, server);
Save	Text that appears on an object in a window	Click on the Save button.

TABLE P-1 Typographic Conventions (*Continued*)

Typeface or Symbol	Meaning	Example
<your text>	Replace the angle brackets and the text they contain with your text.	Edit the file <FWDIR>\lib\yourfile.xx
.	Lines of data or code omitted from example	line 1 line 2 . . . line n
[item]	The item is optional.	dir [/o]
[item1] ... [item2]	List of optional items	dir [/o] [/w] [/s]
item1 item2 item3	Choose one of the items.	copy infile1 infile1 + infile2 infile1 + infile2 + infile3 outfile
<i>italic</i>	Specific values will be shown in italics	one of <i>addnet</i> <i>addapp</i>

SCV API Functions

In This Chapter

<i>Overview</i>	<i>page 7</i>
<i>SCV Check</i>	<i>page 7</i>
<i>Desktop</i>	<i>page 9</i>
<i>Programming Model</i>	<i>page 9</i>
<i>Third Party Programming Model</i>	<i>page 9</i>
<i>OPSEC programming model</i>	<i>page 9</i>
<i>SCV API Overview</i>	<i>page 10</i>

Overview

Check Point's OPSEC (Open Platform for Security) integrates and manages all aspects of network security through an open, extensible management framework. Third party security applications can plug into the OPSEC framework via published application programming interfaces (APIs).

This document describes the SCV (Secure Configuration Verification) Specification.

SCV Check

An SCV check is a DLL (Dynamic Link Library) that queries the security aspect of the configuration of a computer. The check produces a boolean value of "verified" or "non-verified". VPN-1/FireWall-1 SecureClient uses SCV checks to determine the overall security configuration of the computer.

There may be several SCV checks installed on a computer running VPN-1/FireWall-1 SecureClient. Each individual SCV check simply reports "verified" or "non-verified" to VPN-1/FireWall-1 SecureClient.

A single SCV check can test many settings.

Example of three checks:

An "Anti-Virus" SCV check would test whether the anti-virus software is currently executing, has boot sector protection on and has the latest signature files.

Each SCV DLL should expose a function returning a boolean value that will be called by VPN-1/FireWall-1 SecureClient. The SCV check can initiate a call to VPN-1/FireWall-1 SecureClient to report changes in the security configuration. In addition, the SCV check can also pop up a message box to the user and send a log to the VPN-1/FireWall-1 SecureClient log file.

The SCV check implementation should be synchronous when VPN-1/FireWall-1 SecureClient queries SCV check DLLs for current security configuration. It is the responsibility of the SCV check to return an indication of the current status before a pre-configured maximum time-out is reached. If this action takes longer than the time-out, the SCV check will not be taken into account.

The SecureClient does a checksum check on each of the SCV DLLs. If the file has been tampered with the client may not be securely verified.

FIGURE 1-1 Performing an SCV Check Overview

Scv Check Overview

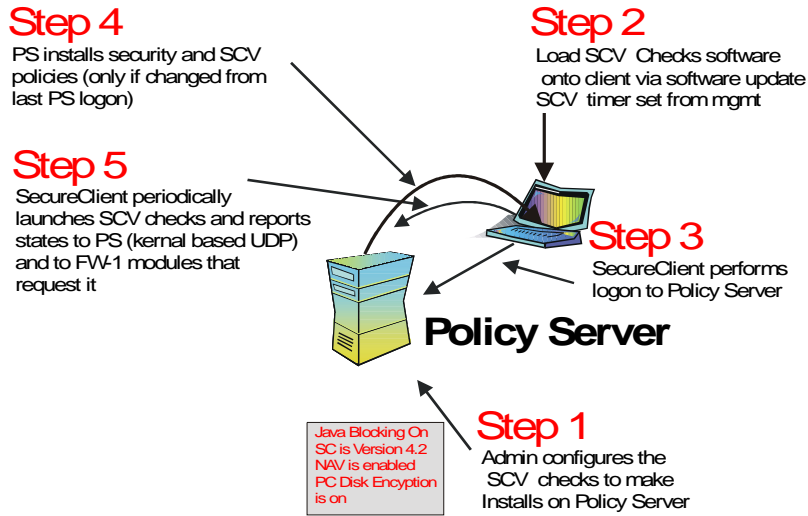


FIGURE 1-1 SCV Check Overview

FIGURE 1-2 Performs SCV Checks and VPN-1/FireWall-1 Intergration

SCV Checks & FW Integration

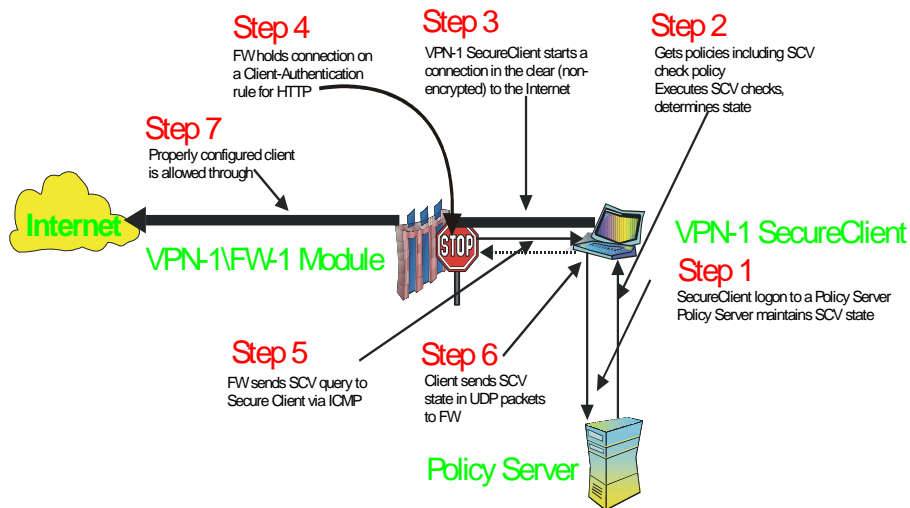


FIGURE 1-2 SCV Checks and VPN-1/FireWall-1 Integration

Desktop

A Desktop is a metaphor for a client computer. Desktops can be PCs or laptops. For instructions on installing Desktop Policies see Chapter 11, section “installing Desktop Policies” of the Check Point Virtual Private Network Manual.

Programming Model

Third Party Programming Model

If an SCV check DLL needs to perform background processing, it should create a thread in which it can perform this processing.

If an SCV check DLL needs to maintain data during its operation it should keep a static data structure.

OPSEC programming model

This is a detailed description of the usage and integration of SCV checks by VPN-1/FireWall-1 SecureClient.

Aspects of Policy Server Downloads

The SCV Specification has two aspects:

- Downloading an SCV policy to the Desktop
- OPSEC Interface, or the SDK for creating the SCV check DLL

Downloading an SCV Policy to the Desktop

VPN-1/FireWall-1 SecureClient downloads Desktop policies from the Policy Server (VPN-1/FireWall-1 SecureClient RuleBase, SCV Policy, Install Policy and Log Policy). The Policy Server has no means of enforcement and is only a mediator between the management server and VPN-1/FireWall-1 SecureClient.

VPN-1/FireWall-1 module enforces the VPN-1/FireWall-1 rule base for inbound and outbound traffic. It is possible to configure client encryption and client authentication rules to enforce SCV status for clients connecting using these rules. When the VPN-1/FireWall-1 module encounters a connection that requires SCV verification, it can query the connecting machine for its SCV status. VPN-1/FireWall-1 Secure client machines that report a “verified” security configuration will be allowed to connect over these rules, otherwise the connections will be dropped. The VPN-1/FireWall-1 module is only concerned with whether the connecting machine has the right configuration.

OPSEC Interface

The SCV OPSEC interface defines the means by which a third party vendor can write its own SCV checks which will verify the desktop configuration. Once a vendor wants to interact with the VPN-1/FireWall-1 SecureClient on the desktop the third party DLL needs to be installed on the Desktop. The administrator adds this installation to the SCV policy that is downloaded from the Policy Server, which instructs VPN-1/FireWall-1 SecureClient to enforce the new SCV check.

Enforcement of SCV checks

Once the SCV check appears in the SCV Policy file it needs to be installed on the desktop, otherwise VPN-1/FireWall-1 SecureClient will consider the machine “non-verified”, and it will not be allowed to connect on VPN-1/FireWall-1 rules that enforce SCV checks.

SCV security configuration is maintained by VPN-1/FireWall-1 SecureClient at all times. If an SCV check detects a change in the security configuration, it can report this change to VPN-1/FireWall-1 SecureClient and the new SCV status will take effect immediately.

SCV API Overview

The API describes the method for building SCV checks via OPSEC SCV API. It also describes the SCV policy files that should be installed on the management server. Furthermore it details installation procedure and registry configuration of an SCV check on the client machine.

SCV Test Tool

This is a tool which enables third party SCV Vendors to check their SCV DLL without using VPN-1/FireWall-1 SecureClient.

SCV API Functions

In This Chapter

<i>Overview</i>	<i>page 11</i>
<i>OPSEC Files</i>	<i>page 11</i>
<i>OPSEC Table of Libraries</i>	<i>page 12</i>
<i>General API's</i>	<i>page 12</i>
<i>Call Back Functions</i>	<i>page 15</i>
<i>Configuration</i>	<i>page 18</i>
<i>How to create the DLL</i>	<i>page 18</i>
<i>Integration with VPN-1/FireWall-1 SecureClient</i>	<i>page 19</i>
<i>SCV Policy Description</i>	<i>page 20</i>
<i>SCV Check Tool</i>	<i>page 21</i>
<i>SCVEditor</i>	<i>page 23</i>

Overview

This section describes the functions provided by the OPSEC SCV API. The function prototypes are described in this document.

OPSEC Files

Header files necessary for the SCV OPSEC API.

TABLE 2-1 List of SCV Header Files

file name	description
Scv_Api.h	contains the functions used to communicate with the user and the VPN-1/FireWall-1 SecureClient
Scv_callback.h	contains the functions that must be implemented by the 3rd party
Scv_error.h	contains the error code conventions
Scv_Internals.h	contains the internal file that must be included in the user implementation

OPSEC Table of Libraries

You must statically link the following libraries into SCV DLL in order to interact correctly with VPN-1/FireWall-1 SecureClient

TABLE 2-2 List of Check Point SCV Interface Libraries

library name	description
PiLib.lib	includes the interface that binds 3rd party code to SCV
Sysprox.lib	includes binding to the VPN-1/FireWall-1 SecureClient
Vertlator.lib	includes a version translation mechanism
Register.lib	includes an auto registration mechanism of the SCV PLL into the registry of VPN-1/FireWall-1 SecureClient

General API's

The general API's may be used as needed by the SCV DLL.

TABLE 2-3 General SCV API's

<i>UserMessageBox</i>	<i>page 12</i>
<i>LogScv</i>	<i>page 12</i>
<i>NotifySCVStatus</i>	<i>page 13</i>
<i>UserAllocateString</i>	<i>page 13</i>
<i>ImpersonateUser</i>	<i>page 14</i>
<i>RevertSelf</i>	<i>page 14</i>
<i>IsUserLoggedIn</i>	<i>page 14</i>

UserMessageBox

UserMessageBox creates a WIN32 message box pop-up for the user at any time.

Prototype

```
SCV_STATUS UserMessageBox (char * lpText, char * lpCaption, unsigned int uType);
```

Arguments

TABLE 2-4 UserMessgeBox Arguments

argument	meaning
lpText	text to appear in the message box
lpCaption	message box title
uType	window type win32 message box options such as: MB_OK, etc.

Return Values

SCV_STATUS as defined in SCV_error.h

LogScv

LogScv creates a log entry which will be sent to the log server via the Policy Server.

Prototype

```
SCV_STATUS LogScv (char* Origin, char* LogMessage, int Alert);
```

Arguments

TABLE 2-5 LogScv Arguments

argument	meaning
Origin	SCV check name
LogMessage	string with log message
alert	if value of alarm is 1 log is of type alert, if value is 0 log is normal (alerts may be implemented in future versions of VPN-1/FireWall-1 SecureClient)

Return Values

SCV_STATUS as defined in SCV_error.h

NotifySCVStatus

NotifySCVStatus notifies VPN-1/FireWall-1 SecureClient of a change in the SCV status as detected by the SCV DLL. This function is most commonly used asynchronously, from a separate thread maintained by the SCV DLL which periodically makes a configuration test and sends NotifySCVStatus (FALSE) if the configuration failed.

Prototype

```
SCV_STATUS NotifySCVStatus (BOOL STATUS);
```

Arguments

TABLE 2-6 NotifySCVStatus Arguments

argument	meaning
STATUS	true the configuration is “verified”, false the configuration is “non-verified”

Return Values

SCV_STATUS as defined in SCV_error.h

UserAllocateString

UserAllocateString allows the SCV DLL to allocate a buffer in which to store the SCV name.



Note – You can allocate the SCV name buffer. GetScvRegistrationParams expects to receive a pointer to this buffer.

Note – This API is restricted to the scope of GetScvRegistryParams **do not** use it in other scopes.

Prototype

```
SCV_STATUS UserAllocateString (int StringSize, char ** AllocatedPointer)
```

Arguments

TABLE 2-7 UserAllocateString Arguments

argument	meaning
StringSize	required buffer size (including the null terminating character)
AllocatedPointer	returned for usage in GetScvRegistrationParams

Return Values

SCV_SUCCESS on success, SCV_ILLEGAL_STRING_SIZE or SCV_ALLOCATION_FAILED on failure.

ImpersonateUser

`ImpersonateUser` allows the calling thread to impersonate the security context of a logged-in User.

Note – Before running `ImpersonateUser`, run `IsUserLoggedIn` to check to see if the user logged.

Note – Registry manipulations:

The `ImpersonateUser` API uses WIN32 API's, and since the SCV DLL runs in a service context (under `SR_Service.exe`), it is recommended to use the actual `HKU\SID` in order to impersonate an active user registry rather than closing the `HKEY_CURRENT_USER` registry key, as this action may cause a race condition on the User's hives.



For further information see:

<http://support.microsoft.com/support/kb/articles/Q199/1/90.ASP>

For sample code see:

<http://support.microsoft.com/support/kb/articles/Q168/8/77.ASP>

Note – Supported on all WIN32 platforms but has no meaning on W9X.

Prototype

```
SCV_STATUS ImpersonateUser();
```

Arguments

none

Return Values

`SCV_SUCCESS` on success, `SCV_FAILED_TO_IMPERSONATE` on impersonation failure or `SCV_NOT_IMPLEMENTED` if not implemented.

RevertSelf

`RevertSelf` terminates the impersonation of a client application.



Note – Supported on all WIN32 platforms but has no meaning on W9X.

Prototype

```
SCV_STATUS RevertSelf();
```

Arguments

none

Return Values

`SCV_SUCCESS` on success, `SCV_FAILED_TO_REVERT` on revert failure or `SCV_NOT_IMPLEMENTED` if not implemented.

IsUserLoggedIn

`IsUserLoggedIn` lets the calling thread the information whether user logged on and SC GUI is up.

Prototype

```
SCV_STATUS IsUserLoggedOn(BOOL * bIsActive);
```

Arguments

TABLE 2-8 IsUserLoggedOn Arguments

argument	meaning
bIsActive	Returns TRUE if user logged on and SC GUI is up. Otherwise FALSE.

Return Values

SCV_SUCCESS on success, SCV_FAILED_TO_GET_STATE on failing fetch logged on state, SCV_NOT_IMPLEMENTED if not implemented.

Call Back Functions

VPN-1/FireWall-1 SecureClient may call these functions, they must all be implemented in the SCV DLL.

TABLE 2-9 SCV Call Back Functions

<i>GetScvRegistrationParams</i>	<i>page 15</i>
<i>Start</i>	<i>page 16</i>
<i>Stop</i>	<i>page 16</i>
<i>Init</i>	<i>page 16</i>
<i>Clean</i>	<i>page 17</i>
<i>Status</i>	<i>page 17</i>
<i>GetScvDiagnostics</i>	<i>page 17</i>

GetScvRegistrationParams

GetScvRegistrationParams is called by the automatic registration mechanism (Pireg.exe) to register or de-register the SCV check into the registry.

Prototype

```
GetScvRegistrationParams (char**vPiName, DWORD *dwMajorVersion, DWORD *dwMinorVersion, char **vDisplayName, char **vszPrivateData, int install);
```

Arguments

TABLE 2-10 GetScvRegistrationParams Arguments

argument	meaning
vPiName	returns SCV check name Note – vPiName is a unique name that represents the SCV dll which is enforced by SecureClient through SCV policy (“SCV Policy Example (local.scv)” on page 20).
dwMajorVersion	returns SCV check major version number
dwMinorVersion	returns SCV check minor version number

argument	meaning
vDisplayName	displayed SCV name Note – vDisplayName should contain a short description of the SCV name and functionality to be displayed by the SecureClient Diagnostics.
vszPrivateData	private data (usage to be determined)
install	1 to register SCV check, 0 to de-register SCV check

Return Values

error code SCV_STATUS on success, SCV_GENERAL_FAIL on failure.



Note – SCV API's: UserMessageBox, LogScv and NotifySCVStatus should not be called in the above callback scope.

Start

Start is called when the SCV check is started. After Start is called the SCV DLL can send SCV status to the VPN-1/FireWall-1 SecureClient. Start will be called after Init is called.

Prototype

```
SCV_Status Start(int argc, char ** argv);
```

Arguments

TABLE 2-11 Start Arguments

argument	meaning
argc	the number of arguments in argv
argv	an array of string arguments in the form "argname=argvalue", which are the parameters provided in the local.scv file for the SCV plugin DLL (see parameters section in local.scv sample). argv[0] is the SCV check name.

Return Values

error code SCV_STATUS on success, SCV_GENERAL_FAIL on failure.

Stop

stop is called when VPN-1/FireWall-1 SecureClient stops usage of a SCV DLL. After stop is called SCV status should not be sent to VPN-1/FireWall-1 SecureClient.

Prototype

```
SCV_STATUS Stop ();
```

Arguments

none

Return Values

error code SCV_STATUS on success, SCV_GENERAL_FAIL on failure.

Init

Init is the initialization function for SCV DLLs. It can be used for allocation and initialization.

Prototype

```
SCV_Status Init(void *Reserved);
```

Arguments

TABLE 2-12 Init Arguments

argument	meaning
Reserved	N/A

Return Values

error code `SCV_STATUS` on success, `SCV_GENERAL_FAIL` on failure.

Clean

Clean is the function that unloads SCV DLLs. It can be used for de-allocation.

Prototype

```
SCV_STATUS Clean();
```

Arguments

none

Return Values

error code `SCV_STATUS` on success, `SCV_GENERAL_FAIL` on failure.

Status

Status is called by VPN-1/FireWall-1 SecureClient when it needs current SCV status (“verified” or “non-verified”) from the SCV DLL.

Prototype

```
SCV_STATUS Status();
```

Arguments

none

Return Values

`SCV_CHECK_PASSED` in case the status is “verified” or `SCV_CHECK_FAILED` in case the status is “non-verified”.

GetScvDiagnostics

SCV has new diagnostics features that can be ported to SecureClient. VPN-1/FireWall-1 SecureClient calls `GetScvDiagnostics` when it needs a current SCV rational string, showing secure or insecure configuration from the SCV DLL. Every periodic check SecureClient queries both status callback and `GetScvDiagnostics` for their current state from the SCV DLL.

Prototype

```
SCV_STATUS GetScvDiagnostics (char ** ppDiagnostics);
```

Arguments

TABLE 2-13 GetScvDiagnostics Arguments

argument	meaning
PpDiagnostics	null terminated string



Note – Copy rational string into PpDiagnostics which is a pre-allocated buffer limited to 1024 characters.

Return Value

error code SCV_STATUS on success, SCV_GENERAL_FAIL on failure.

Configuration

How to create the DLL

Open an empty MSDEV project (win32 Dynamic-Link-library).



Note – It is recommended that you use version Visual C++ 6.0 Service Pack 4 or above.

Note – Minimum libraries for compilation on a MSDEV environment using WIN32 is advapi32.lib.

Add the above libraries listed in *page 12* to the project's libraries path. Create a new C file or use one of the sample C files provided and include the following files in it:

```
#include "Scv_error.h"
#include "Scv_Api.h"
#include "Scv_Internals.h"
#include "Scv_Callback.h"
```

Implement the functions defined in Scv_Callback.h. You must implement at least a stub, if the function needs no implementation

Creating DllMain like the one provided in the sample:

```
/*
 * DllMain for DLL startup
 * This section is necessary for SCV Plugin functionality.
 */
BOOL APIENTRY DllMain( HANDLE hModule, DWORD ul_reason_for_call,
LPVOID lpReserved)
{
    /*
     * Initialize SCV Plugin with Desktop framework.
     */
    ContainerInitiator();
    switch ( ul_reason_for_call )
    {
        case DLL_PROCESS_ATTACH:
            break;
```

```

    }
    return TRUE;
}
}.
```



Note – You are responsible for freeing any memory that is allocated, with the exception of the buffer allocate by UserAllocateString.

Compile and build the DLL.

Use /MDd for debugging or /MD for retail

In Msdev **Config Project Settings** enter the **C/C++** tab on the **Code Generation Category** choose **run time library** then choose **Debug Multithreaded DLL** (for debugging purposes) or **Multithreaded DLL** for retail.

Using an SCV DLL with VPN-1/FireWall-1 SecureClient

Place a policy definition file (local.scv) that includes the SCV check and its parameters in the management server's \$FWdir/conf directory. Install a policy on "SecureClient" (this will transfer the local.scv file to policy servers). Using VPN-1/FireWall-1 SecureClient perform log on to a policy server. The local.scv will be downloaded into the SecuRemote/Policy directory. If the DLL is registered with VPN-1/FireWall-1 SecureClient (either manually or via PiReg.exe), the SCV DLL should be loaded and activated.

Integration with VPN-1/FireWall-1 SecureClient

Installation

There are three steps necessary to perform in order to do an installation:

1. Copy the DLL into <SecuRemote\bin directory>
2. Register the DLL using PiReg.exe.
3. Add this SCV check and its parameters into the SCV policy file (local.scv) in the conf directory of the Management Server.

Auto Registration

Use Pireg.exe (provided with the OPSEC SDK)

- To register a new SCV plugin DLL:

Run Pireg.exe <path\Dllname.dll> using command prompt

- To de-register an existing SCV plugin DLL:

Run Pireg.exe -d <path\Dllname.dll> using command prompt



Note – 3rd party vendors should use PiReg to register SCV plugins on SecureClient.

- Checksum calculation is a part of the PiReg registration.
- GetScvRegistrationParams will be called by PiReg.exe.
- This file (PiReg.exe) should be deleted after the DLL is registered.
- You can install the SCV DLL in any path directory on the client.

SCV Policy Example (local.scv)



Note – If you make a mistake in the object file it will result in a corrupted file error (SCV state will be “non-verified”. Using the SCVEditor will eliminate this problem, information can be found about the “SCVEditor” on page 23.)

```

Policy file (SET)
(SCVObject
    :SCVNames (
    : (SCVGroup1
        :type(group)
        :(samplescv1)
        :(samplescv)
    )
    : (SCVGroup2
        :type (group)
        :(emptyscv)
    )
    :(samplescv
        :type (plugin)
        :parameters (
            :n1param1(value1)
            :n1param2(value2)
            :n1param3(value3)
        )
    )
    :(emptyscv
        :type(plugin)
        :parameters (
            :n2param1(value1)
            :n2param2(value2)
        )
    )
)
:SCVPolicy(
    : (SCVGroup1)
)
)

```

SCV Policy Description

The SCVPolicy set contains the groups of SCV checks that should be used. In SCVGroup1 there are two SCV checks defined (samplescv and samplescv1). The first SCV check from SCVGroup1 that is registered correctly will be used by VPN-1/FireWall-1 SecureClient. samplescv and samplescv1 are similar SCV checks in this example, and at least one of them should be used to report SCV status. Since samplescv1 is not defined properly, samplescv

will be used instead. The SCVPolicy does not contain the emptyscv SCV check, therefore it will not be used at all. samplescv contains three parameters which will be passed in the Start function.

SCV Check Tool

The SCV Check Tool is an additional utility provided by the SCV OPSEC SDK. It is implemented in checktool.exe. The checktool is a command line testing tool which enables third parties to run or debug SCV DLL's independently. This tool can be used to check SCV DLLs without the use of VPN-1/FireWall-1 SecureClient.

Activating the Tool

To use the checktool open a DOS Window and type the path and command path\checktool.exe <path\scv DLL name>

Example

```
D:\Temp\checktool.exe C:\samplescv
```

Each of the following functions is called from the menu below:

Checktool Menu

The SCV function of the Init DLL is called

```
D:\Temp\checktool>checktool scvcmd
Choose command number:
1      Init.
2      Start.
3      Get Scv check Status.
4      Stop.
5      Clean.
6      Reset.
7      Init & Start.
8      Stop & Clean.
9      Operate Scv check under robust scenario.
10     Set parameters file path.
11     Load params from file.
100    Exit.
```

FIGURE 2-1 Checktool Menu

Life Cycle of SCV DLL

TABLE 2-14 Checktool Main Menu Description

...type	command name	description	page
1	Init	initializes the Init callback of the SCV DLL	page 16
2	Start	start or Restart the Start callback of the SCV DLL	page 16
3	Get Scv check Status	get Status from the Status callback of the SCV DLL and query GetScvDiagnostics for SCV rational string	page 17
4	Stop	calls the callback that stops the SCV DLL	page 16
5	Clean	calls the callback that cleans the SCV DLL	page 17
6	Reset	reset the checktool to restart checking (can be called at any point of testing)	none
7	Init & Start	call Init and Start one after the other	
8	Stop & Clean	call stop and clean one after the other	
9	Operate Scv check under robust scenario	Runs a simulation which tests a realistic scenario. Should be called only after clean or reset or before init.	
10	Set parameters file path	Direct the checktool to the directory of your params.txt file.	
11	Load params from file	initiate loading of params from the params.txt file	
100	Exit	leave the application	

Sequence for Running Checktool

A logical sequence for running the menu items starting with init is described in FIGURE 2-2 below.

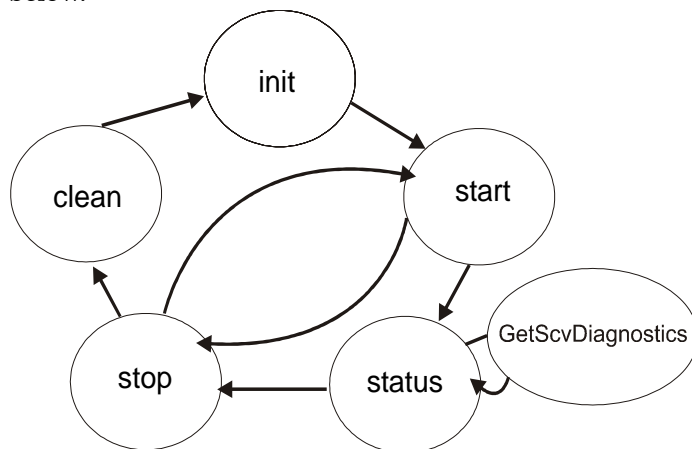


FIGURE 2-2 Testing Flow State Machine

If you don't work in a logical sequence you will get an error and the command will not be performed. After an error you can continue from the same point.



Note – Each time you use the SDK API in the scv DLL a message will appear describing the API.

Passing Arguments to the Start Callback (argc, argv)

In cases where SCV DLL uses parameters you can simulate passing them to your SCV. Create a text file called Params.txt and put your parameters in it. Then direct the checktool to it using menu item 10 (see FIGURE 2-1 above). Using menu item 11, load the parameters. The next time you call menu item 2 the parameters will pass to the SCV DLL via the argc, argv parameters of the Start callback.

Format of Params.txt file

```
Scvname
Param1=value1
Param2=value2
.      .
.      .
```

Example

```
samplescv
nlparam1=value1
nlparam2=value2
nlparam3=value3
```



Note – See SCV Policy Example on page 20 for a file which passes the same values to samplescv in another format which is used in a VPN environment.

Debugging SCV DLL

It is recommended to create an Msdev project for the DLL and run it step-by-step while using the checktool menu.

Steps for creating a debugging environment in Msdev

- 1** Open the **SCV DLL** in **msdev** and change the file type to **“All Files*.*”** to select the DLL.
- 2** In the **debug** tab of the **project settings**, browse and add the checktool.exe as an **Executable for debug session**
- 3** In the same tab add as **Program Arguments** `<path\scvname>`
- 4** Add breakpoints in your SCV DLL code
- 5** Go

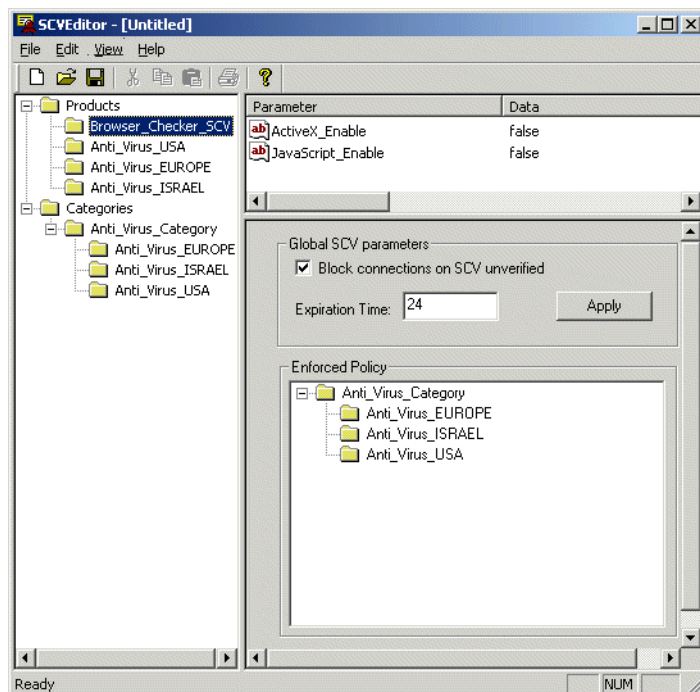
SCVEditor

General information on the policy and its parameters is located in the *“Virtual Private Network”* Book in the *“VPN-1 SecureClient”* chapter.

The SCVEditor is a GUI tool used for the purpose of creating or updating a `local.scv` file. If a **local.scv** file exists, you will be prompted as to whether or not to overwrite the existing **local.scv** file. For more information on local.scv files see the *“SCV Policy Example (local.scv)”* on page 20.

To start SCVEditor From the Explorer Window **open Programs > Check Point SCV SDK > SCVEditor**. The window shown in FIGURE 2-3 should appear:

FIGURE 2-3 SCVEditor Main Window



Overview of the SCVEditor

FIGURE 2-3 can be broken down into three key areas: display tree, product parameters and enforcement

Display Tree—on the left side of the SCVEditor the Display Tree displays products, categories and their sub-directories in a hierarchy. Group in local.scv is a category. First Products are displayed then below Products, Categories are displayed.

Product Parameters—on the top right side of the SCVEditor, the Product Parameters for each product are displayed individually.

Enforcement—from the bottom of the SCVEditor Global SCV parameters are configured. When selections have been made for both **blocking connections on SCV unverified** and **Expiration time**, click **Apply** to refresh the window.

For more information on Enforcement see the “VPN-1 SecureClient” chapter in the “Virtual Private Network” Book.

Specific Commands in the SCVEditor

Create Empty Policy—creates a new empty policy

from **File > Create Empty Policy**, click the **Yes** button to acknowledge the message “**new policy will destroy current policy, continue**”.

Load Policy—loads a pre-existing policy (**local.scv** file)

from **File > Load Policy**, click the **Yes** button to acknowledge the message “**new policy will destroy current policy, continue**”. An open window appears in the Explorer. Select a **local.scv** file and click **open**. The loaded database appears beneath **Products** in the Database Tree.

Generate Policy File—creates a new **local.scv** file

Once Products, Categories and Enforced Policy are set, generate a new policy by selecting **File > Generate Policy File**.

Generate Policy File As—opens a **Save As** window in order to create a new **local.scv** file

Once Products, Categories and Enforced Policies are set, generate a new policy by selecting **File > Generate Policy File As** if you wish to save the **local.scv** file to a different directory.

Exit—exit SCVEditor, you will be prompted as to whether or not to save the **local.scv** file.

SCV Product

An SCV Product is a dll which is used for product verification. SCV Products are the equivalent of SCV plug-in dlls which are represented as type `plugins` in the `local.scv` file.

A minimum condition for SCV is having all products which are part of the enforced policy include one of these products for each enforced category. The first round of each category is enforced.

Add—adds a product

While highlighting Products from the Display Tree, **right click** and select **Add** or from the **Edit** menu select **Product**.

Enter a product name and click **OK**.

Delete—deletes a product

While highlighting a product in the Display Tree, **right click** and select **Delete** or from the **Edit** menu select **Product Delete**.

Click **Yes** to the message “**do you want to delete** product name **product**”.

Modify—modify a product name

While highlighting a product in the Display Tree whose name you want to modify, **right click** and select **Modify** or from the **Edit** menu select **Product Modify**.

Enter a new product name and click **OK**. The new product name replaces the old product name in the Display Tree.

Enforce—enforce a product

While highlighting a product in the Display Tree that you want to enforce, **right click** and select **Enforce** or from the **Edit** menu select **Product Enforce**.

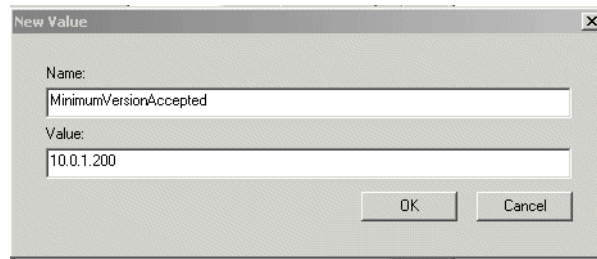
Remove Enforcement—remove an enforced product

While highlighting a product in the Display Tree whose enforced product you want to remove, **right click** and select **Remove Enforcement** or from the **Edit** menu select **Remove Enforcement**.

Add a Value to a Product —add a new value to a product

right click in the Product Parameters window and select **New**. A New Value window, as shown in FIGURE 2-4 will appear:

FIGURE 2-4 New Value for an Empty Policy



Enter the **Name** and **Value** you wish to create in your empty policy and click **OK**.

SCV Category

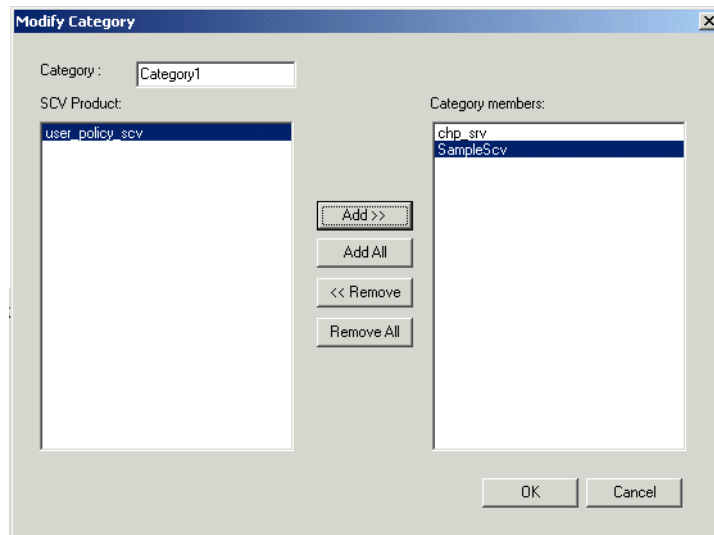
Category refers to the group in the local.scv file.

Add— adds a category

While highlighting Categories from the Display Tree, **right click** and select **Add** or from the **Edit** menu select **Category > Add**.

Enter a category name and click **OK**. FIGURE 2-5 appears.

FIGURE 2-5 Modify Category



Beside **Category**, name the category. Highlight SCV Products which should belong to the newly named category and click **Add**. SCV Products in the left window will become Category Members in the right window. Alternately, if all members should be included as Category Members, click **Add All**. Clicking **OK** creates the category.

Remove a Category Member individually by highlighting the member in the right window and clicking **Remove**. To remove all Category Members click **Remove All**. Clicking **OK** saves changes made to the category.

Delete— deletes a category

To remove members within a category, see **Modify** below.

While highlighting a Category in the Display Tree, **right click** and select **Delete** or from the **Edit** menu select **Category Delete**.

Click **Yes** to the message “do you want to delete categoryname category”.

Modify— While highlighting a category from the Display Tree, **right click** and select **Modify** or from the **Edit** menu select **Category > Modify**. FIGURE 2-5 above appears.

Highlight SCV Products to modify and click **Add or Remove**. Alternately, if all members should be added or removed, click **Add All** or **Remove All**. Clicking **OK** modifies the category.

Enforce— enforce a category

While highlighting a category in the Display Tree that you want to enforce, **right click** and select **Enforce** or from the **Edit** menu select **Category Enforce**.

Remove Enforcement— remove an enforced category

While highlighting a category in the Display Tree whose enforced category you want to remove, **right click** and select **Remove Enforcement** or from the **Edit** menu select **Remove Enforcement**.

Global SCV Parameters

Global SCV Parameters effect each enforced Policy. For more information on Global SCV Parameters refer to the “*Virtual Private Network*” Book in the “*VPN-1 SecureClient*” chapter in the table “*Attributes of the local.scv file*”.

Index

Numerics

3rd party vendor, 9

A

Aspects of Policy Server
 Downloads, 9
Auto Registration, 19

C

Call Back Functions, 15
checktool.exe, 21
Configuration, 18

D

Debugging SCV dll, 23
Desktop, 9
Downloading Scv Policy to the
 Desktop, 9

E

enforcement, 9
Enforcement of SCV checks, 9

G

GetScvRegistrationParams, 15

H

How to create the DLL, 18

I

ImpersonateUser, 14
Init, 16
Installation, 19
Integration with VPN-1
 SecuRemote, 19

IsUserLoggedIn, 14

L

LogScv, 12

M

Modify SCV Editor
 category
 modify, 27

N

NotifySCVStatus, 13

O

OPSEC, 7
OPSEC Interface, 9
OPSEC programming model, 9

P

Params.txt file, 23
 example, 23
PiReg, 19
PiReg.exe, 19
Policy server, 9
Programming Model, 9

R

RevertSelf, 14

S

Scv, 7
Scv API Overview, 10
Scv Check definition, 7
SCV Check Tool
 activating, 21

 life cycle of SCV dll, 22
 menu, 21
 sequence for running, 22
SCV checks for 3rd parties, 9
SCV Editor
 an overview, 24
 category
 add, 26
 delete, 26
 enforce, 27
 enforcement removal, 27
 modify, 27
 policy
 create empty, 24
 generate, 25
 generate as, 25
 load, 24
 product
 add, 25
 add a value to a product, 25
 delete, 25
 enforce, 25
 enforcement removal, 25
 modify, 25
SCV Policy Description, 20
Scv Policy Example (local.scv), 20
SCV Specification has two aspects, 9
Scv Test Tool, 10
Secure Configuration Verification, 7
Start, 16
Start Callback, 23
 format of Params.txt file, 23
 passing arguments, 23
Status, 17
Stop, 16

T

Third Party Programming Model, 9

U

User_Allocate_String, 13

UserMessageBox, 12

Using an Scv dll

DS and PS Configuration, 19

V

VPN-1/FireWall-1 VPN-1

GatewayTM, 9