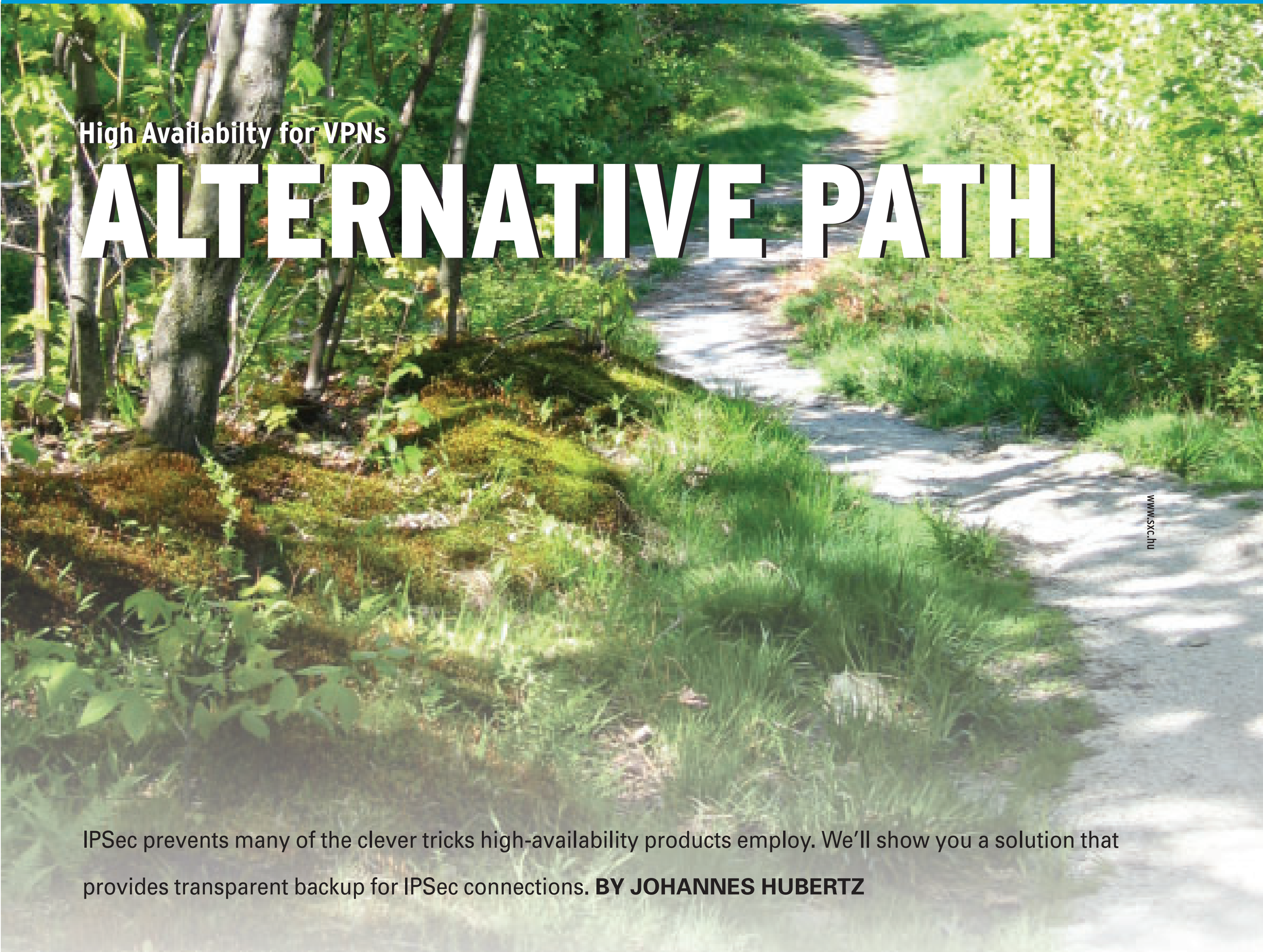

High Availabilty for VPNs

ALTERNATIVE PATH

IPSec prevents many of the clever tricks high-availability products employ. We'll show you a solution that provides transparent backup for IPSec connections. **BY JOHANNES HUBERTZ**

System administrators often want a network connection system that switches transparently to a backup if the primary connection goes down. But if you use a VPN with IPsec to protect your traffic en route through the Internet, the backup line needs some special attention.

The reason for this attention is that IPsec [1] [2] requires consistent IP addresses at the endpoints of a tunnel, so when the network switches to a different tunnel, the IP addresses must switch to the new endpoints or else existing connections will be terminated. The Border Gateway Protocol (BGP [3]) offers a reliable means of maintaining a highly-available pool of IP addresses with a number of providers. Unfortunately, provider service agreements often prevent admins from using BGP for an existing Internet connection.

As a workaround, many admins do without automation, and if worst comes to worst, switch manually from the standard line to the backup line in a process that often involves physically patching the interfaces. Not exactly state-of-the-art. It would be preferable for the network devices to detect a line failure and switch automatically. The ideal system would also automatically generate the settings for both endpoints by referring to a central configuration.

For firewalls and IPsec gateways, the central configuration is state-of-the-art technology. In Linux, SSPE (Simple Security Policy Editor, [4]) handles this. However, the HA solution introduced in this article is not SSPE capable as yet.

Linux-HA

The Linux-HA project [5] is dedicated to high availability solutions for Linux servers. This software allows admins to set up a highly-available VPN (that doesn't require BGP) that will switch from standard to backup operations more quickly. To implement this solution, you need two independent parallel tunnels; you'll only use one of these tunnels at any given time. Each tunnel in each network has its own endpoint, which serves as a gateway to the local network. Linux-HA implements automatic reconfiguration of IP addresses to support this setup.

The HA nodes both have an individual address and a shared address. The shared address is only used by one of the machines at any given time. The mechanism is designed for operating (Web, email…) servers with a shared address. The service runs on both machines and listens to each IP address, however, requests only arrive via the shared IP. This allows Linux-HA to assign the external address to the second machine in case of emergency, ideally without users noticing that the first machine is down.

A serial cable provides the heartbeat for the two machines. The heartbeat is an important part of Linux-HA, as the computers involved use it to check the availability of their peers. If one computer fails, the partner computer adopts the IP address belonging to the first computer. It generates ARP broadcasts using the shared IP and its own MAC address on the LAN (and thus performs a kind of legal ARP-Spoofing). Additionally, the machine enables an interface alias. As soon as the first node gets back online, the heartbeat protocol ensures that Server 2 will disable the alias interface again, while Server 1 enables its interface and ARP broadcasts on the LAN.

Two Linux-HA Installations

The scenario shown in Figure 1 displays the customer on the left, and the outsourcing company on the right. Customer side (left, *gw-aa*) and service provider side IPsec gateways (right, *gw-ba*) use ESP (Encapsulating Security Payload, an IPsec protocol) to send packets across the tunnel. The replacement tunnel (2) is configured on the two backup routers at the top (*gw-ab*, *gw-bb*)in a similar way to Tunnel 1.

Linux-HA is running between *gw-aa* and *gw-ab* and between *gw-ba* and *gw-bb*. The heartbeat uses the serial port, which is not required for any other purpose and is thus independent of the network, IPsec, and IPtables. Both HA installations work separately and independently of one another. In normal operations, *gw-aa* and *gw-ba* have local router IPs (10.1.255.254 left and 10.31.0.254 right). In case of an outage, these addresses migrate to *gw-ab* and *gw-bb*. To allow admins to explicitly connect to the computers in an HA pool, both additionally have static addresses on their respective LANs; this is 10.31.0.252 for *gw-bb* for example.

Scripting an HA-VPN

The default gateways *gw-aa* and *gw-ba* run a shell script like the one shown in Listing 1. The script is launched by an *inittab* entry and monitors the accessibility of the other side, independently of the IPsec tunnel. To do so, the script sends short UDP packets to the echo port. Echo ping is a standard component for most Linux distributions and can use UDP if you set the right options. This approach avoids issues with over zealous firewalls that drop any ICMP messages and thus block normal pings.

As long as the connection between *gw-aa* and *gw-ba* is up, the *FAIL* counter

Listing 1: HA VPN Supervisor

```
#!/bin/bash
# HA VPN Supervisor on gw-aa

# The other end of the tunnel
TARGET="gw-ba"

# Number of seconds between pings
TIMEOUT=1

# Wait MAXFAIL * TIMEOUT, before enabling backout
MAXFAIL=5

# Wait HYSTERE * TIMEOUT after end of outage
# before the script switches back to normal operations
HYSTERE=180

# Assumption: No error condition on start
FAIL=0

VERBOSE=""

ACTION_FAIL_START="/root/bin/HA-VPN-action-script start"
ACTION_OK_AGAIN="/root/bin/HA-VPN-action-script stop"

PING=/usr/bin/echoping
LOG="/usr/bin/logger -t HA-VPN"

math () {
   eval echo "\$(($*))"
}

echo "`date +%Y%m%d%H%M%S` `basename $0` starting" | $LOG

while :
do
VAL=`$PING ${VERBOSE} -u -t $TIMEOUT -s 5 ${TARGET} 2>&1`
ERROR=$?
if [ $ERROR -gt 0 ] ; then
    echo "$DAT $ERROR $FAIL $VAL" | $LOG
    # Timeout occurred
    if [ $FAIL -lt 0 ] ; then
        # Another error during the recovery phase
        FAIL=`math $MAXFAIL + 1`
    fi
    if [ $FAIL -eq $MAXFAIL ] ; then
        # Start backout
        :
        FAIL=`math $FAIL + 1`
        echo "$DAT starting backup now: ${ACTION_FAIL_START}" | $LOG
        ${ACTION_FAIL_START}
    else
        if [ $FAIL -lt $MAXFAIL ] ; then
            FAIL=`math $FAIL + 1`
        fi
    fi
else
    # Ping successful
    if [ $FAIL -gt $MAXFAIL ] ; then
        FAIL=`math 0 - $HYSTERE `
    fi
    if [ $FAIL -le $MAXFAIL -a $FAIL -ge 0 ] ; then
        FAIL=0
    fi
    if [ $FAIL -lt 0 ] ; then
        # Wait for hysteresis period before restarting
        echo "$DAT $ERROR $FAIL $VAL" | $LOG
        FAIL=`math $FAIL + 1`
        if [ $FAIL -eq 0 ] ; then
            # Restore normal operations
            :
            echo "$DAT normal again now: ${ACTION_OK_AGAIN}" | $LOG
            ${ACTION_OK_AGAIN}
        fi
    fi
fi
#echo "$DAT $ERROR $FAIL $VAL" | $LOG
sleep $TIMEOUT
done
# never happens:
exit 0
```

will always be zero. If the ping fails, the script increments the *FAIL* variable in line 53, as long as the value is below *MAXFAIL*. If the next ping is ok, the script sets *FAIL* back to zero (Line 62). If the value reaches *MAXFAIL*, line 50 calls *ACTION_FAIL_START* (Listing 2 with the *start* parameter). *ACTION_FAIL_START* disables the local heartbeat, causing the backup gateway to automatically adopt the local router IP.

Waiting for Normal Operations

The infinite loop keeps running, waits for the line to go back up, and waits for echo ping to return a normal value of 0. When the first response arrives after an outage, the line may not be completely stable, so the script waits before reinstating normal operations. It sets *FAIL* to the negative value of *HYSTERE* (Line 59) and increments *FAIL* for each successful ping (line 67). If another error occurs, Line 43 sets the *FAIL* variable to a value greater than *MAXFAIL* – this keeps the system using the backup.

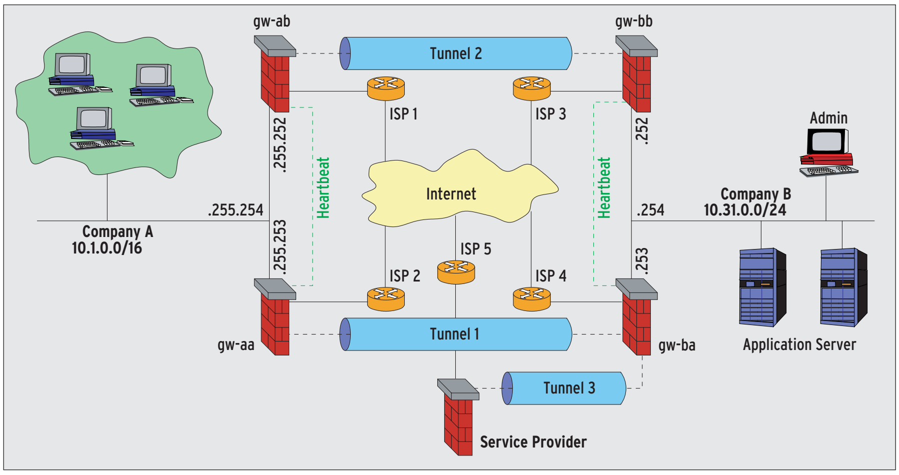

Figure 1: A customer network is attached to an outsourcing service provider via a VPN. This connection uses two alternative paths; if required, Tunnel 2 takes over the role of Tunnel 1.

Normal operations begin when *FAIL* gets back to zero and the script calls *ACTION_OK_AGAIN* in Line 71 (Listing 2 with *stop* parameter). The program also re-enables the heartbeat and thus reinstates the local router IP address.

This approach avoids routing flipflops, where gateways oscillate between tunnels. Internet connections are fraught with short outages, which are resolved after a short period. Three minutes is a good value to ensure stable operations.

Past experience shows that, if the heartbeat at one end of the tunnel downs the gateway, the other end follows suit after a delay of one second at the most.

Listing 2: HA-VPN Action Script

```
#!/bin/bash
# HA-VPN-Aktionsskript

#VERBOSE=-v
VERBOSE=""

NAME=`basename $0`
LOG="/usr/bin/logger -t HA-VPN"

PARAMETER_FAULT=0

if [ $# -ne 1 ] ; then
    PARAMETER_FAULT=1
else
    PARAMETER=$1
    case $PARAMETER in
        start) ;;
        stop)  ;;
        *) PARAMETER_FAULT=1 ;;
    esac
fi

if [ $PARAMETER_FAULT -ne 0 ] ; then
    $LOG " ${NAME}: called with :$*: ==> parameter error, abort"
    echo "`date +%Y%m%d%H%M%S` ${NAME}: parameter error, abort"
    exit 1
fi


ACTION_FAIL_START="/etc/init.d/heartbeat stop"
ACTION_OK_AGAIN="/etc/init.d/heartbeat start"

case $PARAMETER in
    start)  $LOG ${ACTION_FAIL_START} ;
            ${ACTION_FAIL_START} ;;
    stop)   $LOG ${ACTION_OK_AGAIN} ;
            ${ACTION_OK_AGAIN} ;;
esac
exit 0
```

Satisfaction Guaranteed

This solution has been online since 2004 and has repeatedly demonstrated that users do not even notice provider-side router outages. In case of total failure, the normal retry mechanisms used by TCP/IP stacks on the application servers and client PCs can easily bridge the ten seconds before the backup solution cuts in. And the kind of disruption that accompanied outages of this kind previously is now a thing of the past. ■

INFO

[1] RFC 2401: *http://www.ietf.org/rfc/rfc2401.txt*

[2] Freeswan: *http://www.freeswan.ca/code/super-freeswan/*

[3] RFC 1745: *http://www.ietf.org/rfc/rfc1745.txt*

[4] SSPE: *http://sspe.sourceforge.net*

[5] Linux-HA: *http://www.linux-ha.org*

THE AUTHOR

Johannes Hubertz has been a Linux fan since 0.99.2, mainly because Linux allows him to get down to bit level if he wants to. On a professional level, Johannes has had fun managing a few dozen machines for one of Europe's biggest IT enterprises since 1996.