

Inhalt:
Traffic Accounting

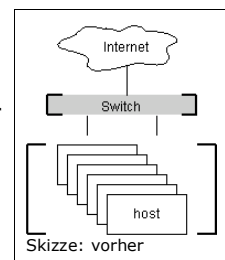
- ▷ Ansätze
- ▷ transparente Realisierung - bridge
- ▷ failover- STP
- ▷ Debugging

Aufgabe:

Nachträglich, in ein bestehendes Netzwerk in einem NOC¹, eine redundante IP Traffic Accounting Lösung zu integrieren. Anforderungen: nachvollziehbare Abrechnungsgrundlage für Endkunden, Anbindung max. 100Mbit/s, Anbindung aktuell 10Mbit/s mit genutzt ca. 3Mbit/s im Peak.

Ansätze:

1. managebare Network Switches einsetzen, Traffic Sammlung mit Hilfe von SNMP
2. kopieren aller Pakete auf einen monitoring Port, Accounting mit Netzwerksniffer
3. Firewall im Datenstrom, oder Router im Datenstrom
4. transparente Bridge im Datenstrom



Lösung 1: SNMP am Switch

- beim Wechsel eines Switch Ports geht die Zuordnung zum Kunden verloren
- wenn Accounting von extern passieren soll muss SNMP auf den Switch möglich sein, d.h. offizielle IP Nutzung, Firmware Security Updates müssen zeitnah eingespielt werden, Standard Passwörter von Switchen problematisch, DoS Attacken evt. auf das Accounting Device möglich, von einigen Switch Herstellern nicht empfohlen Ihre Switches ohne Firewalls im Internet zu betreiben
- + schnell zu Implementieren
- + opensource (RRD / MRTG) Komponenten vorhanden (MRTG – <http://people.ee.ethz.ch/~oetiker/webtools/mrtg/>)
- + kommerzielle Software Produkte vorhanden (z.B. PRTG – <http://www.paessler.com/prtg/>)

Lösung 2: monitoring Port

- setzt entsprechende Switch Infrastruktur voraus
- wiederholt security Probleme mit tcpdump, das eine Analyse der Packetinhalte durchführt, evt. durch bestimmte Pakete DoS Attacken anfällig, kommerzielle Produkte sind mir nicht bekannt
- + mit opensource Komponenten realisierbar, ausserdem auch noch gleich Intrusion Detection möglich

Lösung 3: routing Firewall / Router

- Routing Firewalls ebenso wie native Router, erfordern eine Anpassung aller teilnehmenden Hosts und des Routings des NOC. Danach kann z.B. per iptables Accounting stattfinden
- schwierig (und teuer) redundante Systeme zu installieren, ARP Übernahme muss realisiert werden

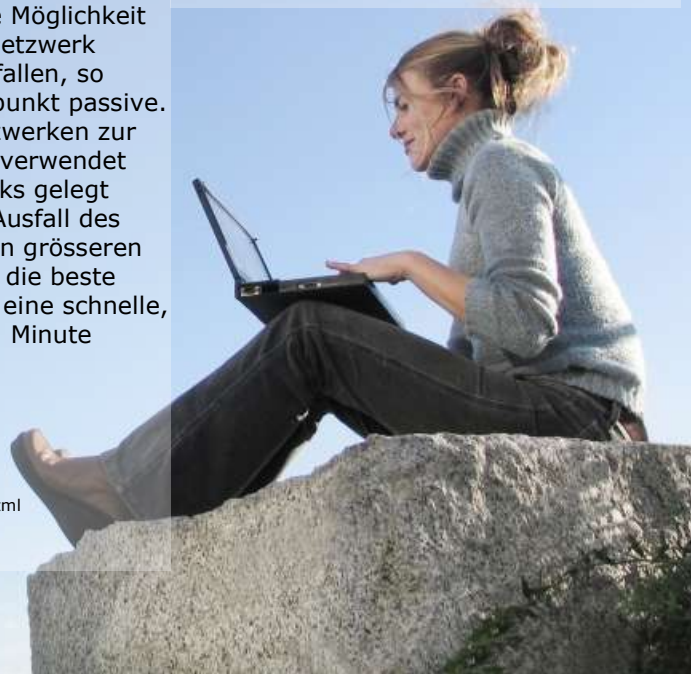
Lösung 4: Linux als Bridge

Linux als Bridge. Linux unterstützt im Kernel den Einsatz als Bridge (im Prinzip ist ein Switch eine multiport Bridge), d.h. Linux kann selbst zum aktiven Netzwerk Switch werden, nur dass durch einen Kernel Patch² Pakete für iptables / netfilter sichtbar gemacht werden können. Vorteile liegen im Einsatz von Standard Komponenten: Linux Kernel, ebttables Patch, iptables, bridge utils

Durch Verwendung von STP³ wird zusätzlich die Möglichkeit geschaffen, 2 Linux Bridges redundant in das Netzwerk Design einzubauen, sollte eine der Bridges ausfallen, so übernimmt automatisch die bis zu diesem Zeitpunkt passive. Realisiert wird das failover per STP, das in Netzwerken zur redundanten Uplink Verkabelung von Switches verwendet werden kann. Konkret können mehrfache Uplinks gelegt werden und alle bis auf einen bleiben bis zum Ausfall des Masters (forwarding Mode) im blocking Mode. In grösseren Umgebungen ist STP wegen der Latenzen nicht die beste Failover Lösung. In kleinen Umgebungen ist es eine schnelle, kostengünstige Variante, die Failoverzeiten < 1 Minute realisiert.

Wichtig, es gibt auch Probleme mit STP⁴.

Genannt seien exemplarisch:
- DoS Möglichkeit durch Austausch eines Gigabit durch 2 FastEthernet Links
- Sniffen des gesamten Traffics durch umleiten durch ein kompromitiertes System das an 2 Switches hängt.



¹ network operation center
² <http://ebtables.sourceforge.net/>
³ spanning tree protocol
⁴ <http://archives.neohapsis.com/archives/vuln-dev/2002-q2/0316.html>

Output

Umsetzung der Aufgabe:

Es wurde sich für die Lösung 4 entschieden, weil gleichzeitig mit dem Accounting eine transparente Filterung von Paketen "bridgewalling" stattfinden kann. Alle Pakete die iptables / netfilter zählt kann er zusätzlich mit firewall rules behandeln. So werden konkret SNMP (UDP 161) Verbindungen auf die Infrastruktur gedropped.

Für die Umsetzung wurden 2 Server wiederverwendet, die jeweils mit
1x CDROM
3x NIC (Network Interface)
CPU > PII 400
128 MB RAM
ausgestattet sind

Auf Festplatten wurde bewusst verzichtet. Das Betriebssystem ist ein Debian Woody mit Vanilla Kernel 2.4.25 von kernel.org, der mit dem ebttables Patches kompiliert wurde. Das Betriebssystem enthält alle notwendigen Komponenten wie iptables, bridge utils, iptraf, tcpdump, ipacc-ng, openssl, openssl.

Von den 3 Interfaces ist ein Interface als Adminzugang vorgesehen und wird zum remote Zugriff auf die Bridge verwendet. Die beiden zusätzlichen Interfaces werden mit brctl zu einer bridge zusammengefasst und zu beiden Switches verbunden.

Die IP Accounting bridges booten dabei vollständig von CD und kopieren Ihre Live-CD Umgebung (ca. 90MB) ins RAM, danach kann die CD ausgeworfen werden.

Dies ist im Hinblick auf Updates interessant, da die CDs so bereits tagsüber getauscht werden können und nachts nur noch ein reboot stattfinden müsste um z.B. einen neuen Kernel zu aktivieren.

Die Aufgabe des Accountings wurde iptables überlassen, die Suite ipacc-ng leistet dabei gute Dienste um sich um das Auslesen und Aggregieren der rule chain counter nicht per Skript kümmern zu müssen.

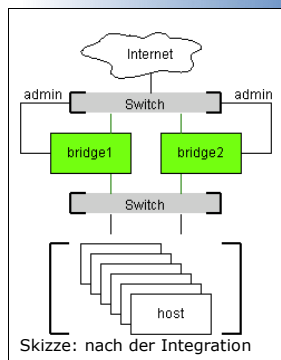
Es werden automatisch alle 5 Minuten die Zähler der Accountingregeln (im Moment pro Server IP incoming und outgoing Traffic) ausgelesen. Die Lösung ist dabei so flexibel, dass auch ein Accounting auf TCP / UDP Port Ebene möglich ist, z.B. HTTP / HTTPS / SMTP. Konkret war dies allerdings nicht gefordert.

Lösung 1, kann dabei als Kontrolle der Implementierung dienen und liefert bis auf kleine Abweichungen identische Werte. Durch die Verwendung der IP Adresse im Accounting und nicht des Switchports bleibt das Setup dabei flexibel und kann leicht auf eine leistungsfähigere Infrastruktur portiert werden.

Die IP Accounting Zahlen erreichen den Kunden stündlich per Mail (SMTP). Dedicated Server Kunden können per lokalem iptables Ruleset die Zahlen selbst kontrollieren.

Es soll nicht verschwiegen werden, dass der Nachteil dieser festplattenlosen Installation darin liegt, dass keine Daten permanent gehalten werden können. Dies wurde aber durch die überwiegenden Vorteile ausgeglichen.

Dem Kunden war die Lösung per E-Mail ausreichend, alternativ sind Szenarios mit Verwendung von z.B. sshfs, ftpfs, nfs oder syslog-ng denkbar, die die Accountingdaten von den Bridges auf einen (oder mehr) Abrechnungsserver zu übertragen.



In dem skizzierten Setup wurde besonders darauf geachtet, keine single point of failures einzubauen, d.h. beide bridges sind autark und können jeweils

den gesamten Traffic bewältigen.

Das gesamte Konstrukt kann dabei jederzeit zur Fehlersuche oder Optimierung mit einem X-Kabel überbrückt werden, wenn auch die Switches am STP teilnehmen.

Die Zeit bis zum Erkennen eines Fehlerzustandes (link-loss) und anschließender Rekonfiguration des Netzes lag, bei Tests, jeweils unter 1 Minute und wurde als ausreichend angesehen.

Hinsichtlich debugging bietet die aufgezeigte Realisierung mit Linux + OpenSource weitreichende Möglichkeiten auch noch im Live-Betrieb sich den Traffic anzuschauen und zu reagieren. z.B. zusätzliche Firewall Rules einzubauen. Tools wie iptraf, tcpdump stehen auf den bridges per ssh Adminzugang zur Verfügung.

Fazit: eine schnelle Implementierung eines transparenten IP Accountings, das keinen SPOF enthält ist möglich. Alle Komponenten sind dabei OpenSource und erlauben Anpassungen an spezifische Kundenwünsche / -hardware.

