System Monitoring

RBG-Seminar 2005/06

Holger Kälberer

20.12.2005

Was ist System-Monitoring?

Überblick

Was überwachen?

Infrastruktur

- Serverraum
- Switches
- traffic

Host-Status

- an/aus
- gesund/kompromittiert
- Resourcen (HD/RAID, CPU, RAM, ...)

Netz/Dienste

- Mail, DNS, http, DB, ...

Woraufhin überwachen?

- Status: gut/schlecht; eher zeitnah; Warnungen rausgeben
- Statistik: traffic/uptime/...; längerfristiger Zustand

Wie reagieren?

- gar nicht/Statistiken erzeugen
- warnen
- automatische Reaktion (IPS)

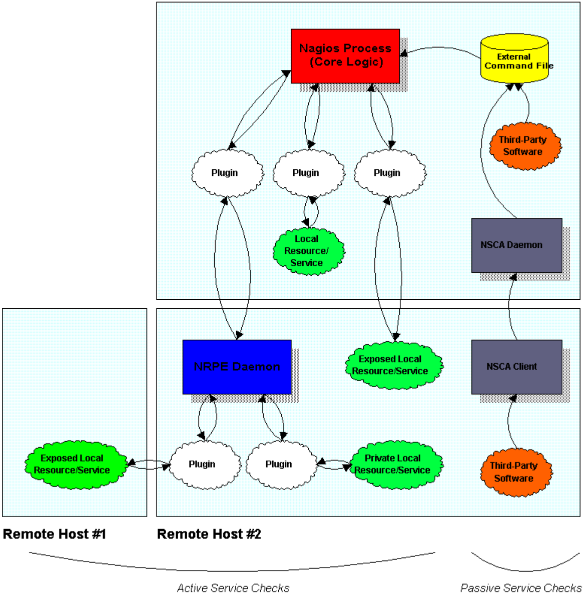
Womit überwachen/Mittel?

- logfiles
- tools zur statistischen Auswertung (mrtg, rrd...)
- IDS-/IPS-tools, portsentry (Unterschied: IDS vs. SM = aussen vs. innen)
- speziell: Monitoring frameworks (snmp, Big Brother, Little Sister, Netsaint/Nagios, ...)

Nagios

Übersicht(1): die wichtigsten Entitäten:

- host, hostgroup, host dependency
- service, servicegroup, service dependency
- contact, contactgroup
- timeperiod
- command
- host/service/hostgroup/servicegroup escalation


Nagios Process (Core Logic)
External Command File
Plugin
Plugin
Plugin
Third-Party Software
Local Resource/ Service
NSCA Daemon
NRPE Daemon
Exposed Local Resource/Service
NSCA Client
Exposed Local Resource/Service
Plugin
Plugin
Private Local Resource/Service
Third-Party Software
Remote Host #1
Remote Host #2
Active Service Checks
Passive Service Checks

Prüfen: checks

Geprüft werden hosts und Dienste
Plugins: ausfuherbare scripte/Programme, die die Tests übernehmen

- direkt vs. indirekt
- aktiv vs. passiv
- leicht zu erstellen
- vielfältige Funktionsweise; vgl. check_fping, check_nrpe, check_cluster2, check_sensors

Ergebnis und Reaktion

Zustände:

- ok vs non-ok; bzw. ok/warning/critical
- soft vs. hard: mehrmaliges Prüfen gegen Fehlalarme; reguliert Benachrichtigungen und event-handlers

Reaktionen:

- Sichern des Ergebnisses (logging/DB)
- Benachrichtigung in Abh. von vielen Faktoren
- event-handlers: Programme, die bei Zustandsänderungen ausgeführt werden

Benachrichtigungen ...

- ... worüber? (warning/critical/unkown/recovery/flapping)
- ... unter welchen Umständen? (notifications enabled? ... für diesen host/service? flapping? innerhalb von timeperiod? ...)
- ... wie oft? (renotifications?)
- ... wen, wann wie? (contacts)

Kontakte und Kontaktgruppen

- bilden Zuständigkeiten ab
- services und hosts sind Kontaktgruppen (contactgroups) zugewiesen
- Kontaktgruppen bestehen aus einzelnen Kontakten (contact)
- Kontakte definieren
  - Medien (email, pager, sms, call, ...), die als Kommandos definiert werden,
  - die relevanten Umstände (critical-states, usw.) und
  - Zeitabschnitte (timeperiods).

Funktionsweise(2): Einige Details

Scheduling

- parallelisiert
- normal_check_interval vs. retry_check_interval, max_check_attempts
- inter-check delay = (average check interval for all services) / (total number of services)
- host-checks werden nur auf Anfrage ausgeführt

Abhängigkeiten

(1) strukturelle Abhängigkeiten: parent-children-Beziehung

(2) host/service-dependencies
- check nur, wenn nicht von Objekt abhängig, das non-ok ist.
- bedingen das Versenden von notifications
- können vererbt werden

Indirekte Tests: nrpe

- check_by_ssh vs. nrpe
- nrpe: Daemon auf client, der seinerseits Kommandos definiert, die der Nagios-Prozess anstossen kann.
- ssl-verschlüsselt
- verteiltes Monitoring durch indirekte Tests (firewall).

Eskalationsstufen

- ermöglichen abgestufte automatisierte Reaktionen über Kommando-Definitionen
- konkret z.B. abgestufte Benachrichtigungen

Weitere Eigenschaften

- Webfrontend
- external commands über named pipe (auch: dynamisches Verändern von Paramtern; adaptive monitorin)
- downtimes
- EPN
- flapping-detection
- Verteiltes Monitoring (dist. server - OSCP - nsca - external command - central server)
- über passive Tests ist die Integration von anderen tools möglich (via nsca; Bsp.: logcheck-Ergebnisse; IDS)
- Konfiguration: modular, template-basiert, ermöglicht Objekt-Hierarchie und Vererbung

Fazit

+ hochgradig konfigurierbar (z.B. commands)
+ hochgradig erweiterbar (Plugins, Medien für notifications, commands)
+ zuverlässig
- komplizierte Konfiguration (z.B.: mehrstufige timeperiods, komplizierte Kommando-Definitionen)
- u.U. schwieriges tracen von Fehlkonfiguration

Logfile-Monitoring

Zentralisiertes Logging

- separate logfiles (Sicherheit!)
- zentralisierte Auswertung
- Archivierung

syslog-ng

- kann auch TCP
- archiviert sources (Netz, lokal)
- via filter (facility/priority; basale Funktionen (host, match, program)
- in destinations (Netz, Dateien, terminals, Programme; macros)

Zeitnahe Auswertung

- syslog-ng
  - Filtern von bestimmten Meldungen ("Authentication failure for illegal user")
  - Reaktion in destinations (denylogin.sh)
- swatch (vielfältiger zu konfigurieren: regexps, Zustand, Reaktion)
- sec (Simple Event Correlator: komplexe Kontextdefinitionen und bedinggte Reaktionen: Wenn 4 mal "Illegal user (S+)" von gleicher IP ...)

Regelmässige Auswertung: logcheck

- Scannen auf vordefinierte Muster (Reguläre Ausdrücke) für
  1. potentielle Risiken bzw.
  2. Fehlalarme
- drei Modi (drei Mustersätze):
  1. paranoid
  2. server
  3. workstation
- drei Sicherheitsstufen und drei Filterebenen
  1. "Security Alerts"
  2. "Security Events"
  3. "System Events"
- Anpassung der Regeln!

Arbeitsweise:

- Logcheck scannt alle angegebenen Dateien
- in Abhängigkeit vom gewählten Modus
- auf den drei Ebenen und
  - warnt bei violations
  - ignoriert Fehlalarme
- abschliessender Bericht
- Index auf schon geprüften Dateien

Fazit

- längere Anpassungsarbeit
+ Verringert des log-Aufkommen enorm
+ anfänglich wählbarer Modus