Monitoring über Naemon im GearFish-Netzwerk

VERSION: 1.0 | LIZENZ: INTERN

GEARFISH TEAM: KEANU AMANN | DAVID KLEINFERCHER

Dieses Dokument beschreibt das Monitoring-Setup auf Basis von Naemon im GearFish-Netzwerk. Es umfasst die Begründung für den Monitoring-Einsatz, die verwendeten Plugins, die relevanten Konfigurationsänderungen sowie die während der Implementierung aufgetretenen Probleme und deren Lösungen.

INHALT

1. Grund für Monitoring

Wegen der teils alten Hardware im Netzwerk erachten wir – Team 12 Gearfish – ein aktives Monitoring als notwendig, um Überlastungen und Störungen im Netz frühzeitig erkennen und gezielt Gegenmaßnahmen einleiten zu können.

2. Was ist Naemon

Naemon ist eine modifizierte Variante der Open-Source-Monitoring-Software Nagios (Naemon = **Na**gios da**emon**) und wurde von ehemaligen Mitentwicklern von Nagios erstellt. Wie Nagios kann Naemon mithilfe von Plugins Dienste und Geräte im Netzwerk überwachen – entweder agent-basiert oder agentless – und nutzt dafür Protokolle wie WMI, SNMP oder SSH. Zur Darstellung der gesammelten Daten kommt das Open-Source-Monitoring-Dashboard **Thruk** zum Einsatz.

3. Plugins

Wir verwenden hauptsächlich die folgenden drei Plugins:

Plugin	**Zweck**
check_alive	Erreichbarkeit per ICMP-Ping
check_by_ssh	Ausführung von Checks via SSH auf Linux-Zielen
check_wmi_plus	Agentless-WMI-Abfragen gegen Windows-Hosts

3.1 CHECK_ALIVE

Pingt das Zielgerät und meldet die Erreichbarkeit zurück.

3.2 CHECK_BY_SSH

check_by_ssh loggt sich mit einem vordefinierten Benutzer per SSH am Ziel ein, führt dort eine Aktion aus (z. B. check_memory, check_uptime, …) und sendet das Ergebnis zurück. Voraussetzung ist, dass die jeweiligen Plugins (etwa check_disk, check_load) auch auf dem Zielgerät installiert sind.

3.3 CHECK_WMI_PLUS

check_wmi_plus führt mit einem vordefinierten Query-User WMI-Abfragen gegen Windows-Geräte aus – agentless. Da wmic deprecated ist, wird zusätzlich der WMIC-Server aus dem **aiowmi**-Projekt benötigt, der die Abfragen des Plugins in CIM-Abfragen übersetzt und das Ergebnis zurückgibt. Der WMIC-Server wird als Service über **gunicorn** eingebunden.

4. Konfigurationsdateien

Im Folgenden sind alle Änderungen an den Konfigurationsdateien dokumentiert.

4.1 /ETC/NAEMON/RESOURCE.CFG

```
# Sets variables to be the path to the plugins
$USER1$=/usr/lib/naemon/plugins
$USER2$=/usr/lib/nagios/plugins
# Sets $USER2$ to be the path to event handlers
#$USER2$=/usr/lib/naemon/plugins/eventhandlers

# Store some usernames and passwords (hidden from the CGIs)
$USER4$=user1
$USER5$=MYSECRETACCESSTOKEN1
$USER6$=kairos
$USER7$=/etc/naemon/sshpass.secret   # Passwort für Kairos liegt in dieser Datei
```

4.2 /ETC/NAEMON/CONF.D/ – ÜBERSICHT

- printer.cfg wurde **gelöscht**.
- switch.cfg wurde in **`firewalls.cfg`** umbenannt.
- Neu/angepasst: windows.cfg, commands.cfg, localhost.cfg, firewall.cfg (siehe unten).

4.3 WINDOWS.CFG

```
define host {
   host_name         winsrvmgmt1
   alias             AD, DHCP and DNS Server
   address           192.168.1.2
   use               windows-server
   hostgroups        windows-servers
}

define host {
   host_name         winbarasrv1
   alias             Baramundiserver
   address           192.168.1.4
```

```
    use             windows-server
    hostgroups      windows-servers
}

define host {
    host_name       winsrvbackup1
    alias           Backupserver
    address         192.168.1.9
    use             windows-server
    hostgroups      windows-servers
}

###############################################################################
# HOST GROUP DEFINITIONS
###############################################################################

# Alle Hosts mit dem Template "windows-server" werden automatisch Mitglied
define hostgroup {
    hostgroup_name  windows-servers
    alias           Windows Servers
}

###############################################################################
# SERVICE DEFINITIONS
###############################################################################

define service {
    service_description Uptime
    hostgroup_name  windows-servers
    use             generic-service
    check_command   check-uptime

}



define service {
    service_description Ping
    hostgroup_name  windows-servers
    use             generic-service
    check_command   check-host-alive
}

define service {
```

```
    service_description Event_View
    hostgroup_name      windows-servers
    use                 generic-service
    check_command       check-events
}

define service {
    service_description CPU_Load
    hostgroup_name      windows-servers
    use                 generic-service
    check_command       check-cpu
}

define service {
    service_description RAM_Load
    hostgroup_name      windows-servers
    use                 generic-service
    check_command       check-ram
}

define service {
    service_description Storage_Use
    hostgroup_name      windows-servers
    use                 generic-service
    check_command       check-storage
}
```

4.4 COMMANDS.CFG

Die folgenden Custom-Commands wurden ergänzt:

```
##############################################################################
# CUSTOM COMMANDS – t12.lan
##############################################################################

#-----------------------------------------------------------------------
# Windows Check Commands
#-----------------------------------------------------------------------

define command {
    command_name      check-storage
    command_line      $USER1$/check_wmi_plus.pl -H $HOSTADDRESS$ -m checkdrivesize -a "C:" -w 75 -c 90 -u
$USER4$ -p $USER5$
}

define command {
    command_name      check-cpu
    command_line      $USER1$/check_wmi_plus.pl -H $HOSTADDRESS$ -m checkcpu -w 80 -c 95 -u $USER4$ -p
$USER5$
}

define command {
    command_name      check-ram
    command_line      $USER1$/check_wmi_plus.pl -H $HOSTADDRESS$ -m checkmem -w 80 -c 90 -u $USER4$ -p
$USER5$
}

define command {
    command_name      check-events
    command_line      $USER1$/check_wmi_plus.pl -H $HOSTADDRESS$ -m checkeventlog -a "System" --
includedata "_EventType=1" --timebefore 86400 -w 0 -c 0 -u $USER4$ -p $USER5$
}

define command {
    command_name      check-uptime
    command_line      $USER1$/check_wmi_plus.pl -H $HOSTADDRESS$ -m checkuptime -u $USER4$ -p $USER5$
}

#-----------------------------------------------------------------------
# Linux Check Commands
#-----------------------------------------------------------------------
```

```
  define command {
    command_name      check-linux-storage
    command_line      sshpass -f $USER7$ $USER2$/check_by_ssh -H $HOSTADDRESS$ -l $USER6$ -o
StrictHostKeyChecking=no -C "/usr/lib/nagios/plugins/check_disk -w 25% -c 10%"
  }

  define command {
    command_name      check-linux-ram
    command_line      sshpass -f $USER7$ $USER2$/check_by_ssh -H $HOSTADDRESS$ -l $USER6$ -o
StrictHostKeyChecking=no -C "/usr/lib/nagios/plugins/check_memory --available -w 20%: -c 10%:"
  }

  define command {
    command_name      check-linux-cpu
    command_line      sshpass -f $USER7$ $USER2$/check_by_ssh -H $HOSTADDRESS$ -l $USER6$ -o
StrictHostKeyChecking=no -C "/usr/lib/nagios/plugins/check_load -w 1.5,1.0,0.8 -c 3.0,2.5,2.0"
  }

  define command {
    command_name      check-linux-uptime
    command_line      sshpass -f $USER7$ $USER2$/check_by_ssh -H $HOSTADDRESS$ -l $USER6$ -o
StrictHostKeyChecking=no -C "/usr/lib/nagios/plugins/check_uptime"
  }
```

4.5 LOCALHOST.CFG (LINUX)

Host und Hostgruppe wurden angepasst:

```
define host {
    host_name          ubunaemonsrv1
    alias              Naemon Server
    address            127.0.0.1
    use                linux-server
    hostgroups         naemon-servers
}

##############################################################################
# HOST GROUP DEFINITION
##############################################################################



define hostgroup {
    hostgroup_name     naemon-servers
    alias              Naemon Servers
}
```

4.6 FIREWALL.CFG

Host wurde angepasst:

```
define host {
    host_name          t12graetefw1          ; Name dieses Switches/Firewalls
    alias              Main Firewall         ; Langer Name
    address            192.168.1.253         ; IP des Geräts
    use                generic-switch
    hostgroups         firewalls             ; Zugehörige Hostgruppen
}
```

5. Thruk

Feld	**Wert**
Admin-User	thrukadmin
Passwort	BrT53Q@4my€

6. aiowmi / WMIC-Server

Änderungen an wmic_server.yaml:

- Alle Default-User und -Tokens **bis auf** user1 / MYSECRETACCESSTOKEN entfernen.
- Die Credentials von kairos werden anschließend bei user1 ergänzt.

Der WMIC-Server wird als Service über **gunicorn** betrieben und vom check_wmi_plus-Plugin angesprochen.

7. Probleme und Herausforderungen

Im Zuge der Implementierung sind folgende Probleme aufgetreten:

7.1 THRUK-WECHSEL VON APACHE AUF NGINX

Problem: Bei der Erstinstallation war das Thruk-Dashboard fehlerhaft. Eine Anmeldung als `thrukadmin` war nicht möglich, und Thruk interagierte nicht korrekt mit Naemon. Anpassungen an den Thruk-Konfigurationsdateien blieben wirkungslos.

Lösung: Eine Neuinstallation von Thruk mit Umstieg von Apache auf **Nginx** behob sowohl die Anmeldeprobleme als auch die Interaktionsfehler mit Naemon.

7.2 FALSCHES PASSWORT-PARSING

Problem: Trotz korrekt installierter Plugins auf den Zielgeräten schlugen die `check_by_ssh`-Abfragen fehl.

Lösung: Die Ursache war zweigeteilt:

1. Die Variablen in `resource.cfg` wurden nicht korrekt geparst.
2. `useradd` hatte auf dem Zielgerät – trotz `-p` – die Passwörter nicht richtig gesetzt.

Die Passwörter wurden über `passwd` neu vergeben. Außerdem wird das SSH-Passwort jetzt aus `/etc/naemon/sshpass.secret` gelesen, wodurch das fehlerhafte Variablen-Parsing umgangen wird.

7.3 SYNTAX UND PLUGIN-IMPLEMENTIERUNG

Durch die Vielzahl an Plugins und die manuelle Bearbeitung von Konfigurationsdateien kam es mehrfach zu Syntaxfehlern und Tippfehlern in den Befehlszeilen, die teilweise erst nach längerer Suche gefunden wurden. Empfehlung für Folgearbeiten: jedes geänderte Command einzeln mit `naemon -v /etc/naemon/naemon.cfg` validieren, bevor der Dienst neu gestartet wird.

Stand: 2026-05-27.