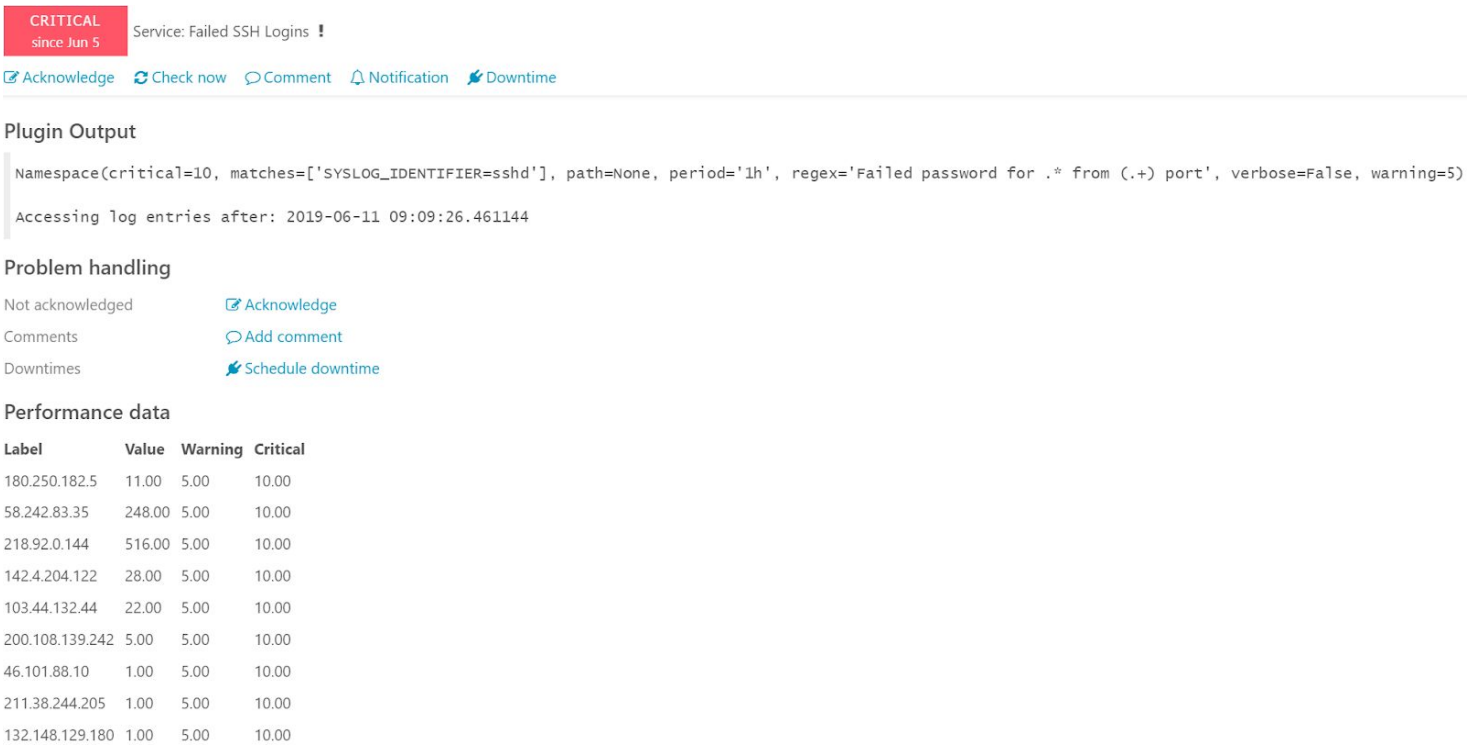
Icinga2 Plugin: Check-Journald



Projekt durchgeführt von:
Siegrist Louis
Gerber Raphael

Betreut durch:
Prof. Dr. Nikkel Bruce

Datum: 13.06.2019

Inhalt

1 Projektbeschreibung

Das Ziel des Projektes es eine Schnittstelle zwischen der Unix “systemd” Komponente “journald” und der Monitoring-Software Icinga2 zu entwickeln. Ziel ist es den “systemd” Journal Inhalt von Icinga2 aus zu überwachen. Sobald diese Schnittstelle entwickelt ist, gibt es diverse Möglichkeiten deren Funktionalität zu erweitern. Die folgende Liste von Möglichkeiten enthält optionale Projektziele und/oder mögliche Themen für weitere Studien im Rahmen einer Bachelor Thesi:

- Implementation Log Analyse: Dies könnte etwas so simples sein, wie das erkennen von bestimmten Fehlermeldungen oder so komplex wie eine Log-basiertes Intrusion-Detection-System (IDS).
- Überwachen anderer “systemd” Komponenten: “systemd” besteht aus vielen weiteren Services (Daemons) die mithilfe von Icinga überwacht werden könnten. Dies reicht von “networkd”, über “logind” bis hin zu User Sessions und Prozess Management. Man könnte die Schnittstelle folgendermassen erweitern, dass sie das Überwachen weiterer solcher “systemd” Komponenten unterstützt.
- “Journald” Bridge: Es gibt eine Vielzahl von forensischen und analytischen Tools auf dem Markt. Eine mögliche Erweiterung zum Projekt könnte die Entwicklung einer allgemein gebräuchlichen “journald bridge” sein, die mit mehreren solchen Tools kompatibel ist.

Ob einige dieser optionalen Ziele Teil des Projektes werden wird davon abhängen, wie gut die Entwicklung der ursprünglich angedachten Schnittstelle voranschreitet. Ob diese Ideen allenfalls als Bachelor Thesis Themen taugen, wird ebenfalls während dem Projekt evaluiert.

Das Projekt wird von den Studenten Siegrist Louis und Gerber Raphael der Berner Fachhochschule durchgeführt. Betreut wird das Projekt durch Prof. Nikkel Bruce.

2 Konzept

2.1 Funktionsweise von Icinga Plugins

Im Grunde genommen sind Icinga Plugins nichts weiter als eigenständige Programme, die in regelmässigen Abständen von Icinga aufgerufen werden. Dabei sind diverse Programmier- und Skriptsprachen unterstützt, wie beispielsweise Bash, Python, Perl, Ruby, PHP, C/C++, GO, etc. Beim Aufruf der Plugins können diverse konfigurierbare Command-Line-Interface (CLI) Argumente mit übergeben werden. Dadurch kann das Verhalten der Plugins durch entsprechende Konfigurationen in Icinga beeinflusst werden. Anschliessend wird der Rückgabewert, sowie der Konsolen-Output der Plugins von Icinga interpretiert. Dabei wird folgendermassen vorgegangen:

- Die erste Konsolen-Ausgabe des Plugins wird von Icinga als Service Statusnachricht interpretiert und im Web-GUI entsprechend angezeigt.
- Alle Konsolen-Ausgaben, die mit dem Pipe-Symbol (“|”) beginnen, werden als sogenannte Performance-Daten interpretiert. Diese werden von Icinga im entsprechenden Bereich des Web-GUIs angezeigt und gegebenenfalls grafisch dargestellt. Für weitere Informationen bezüglich der Performance Daten, siehe Kapitel 2.6 der Monitoring Plugins Development Guidelines [1].
- Der Rückgabewert bzw. Exit-Code des Plugins bestimmt schlussendlich den Icinga Status. Die folgenden numerischen Werte und deren Bedeutungen sind dabei Standard:

Status:	Code:
OK	0
Warning	1
Critical	2
Unknown	3

Wenn das Ausführen des Plugins zu lange dauert und einen gewissen Schwellenwert überschreitet, so wird die Ausführung automatisch abgebrochen und der Status “Critical” mit entsprechender Statusnachricht angezeigt.

2.2 Check-Journald Plugin

Das Check-Journald Plugin, welches im Rahmen dieses Projektes entwickelt wurde, ist in Python geschrieben. Es ermöglicht es dem Benutzer mit Hilfe entsprechender Konfigurationsmöglichkeiten ein Monitoring des systemd Journals einzurichten. Dabei ist die konkrete Ausprägung dem Benutzer überlassen, das Plugin stellt lediglich ein Framework für entsprechende Monitorings zur Verfügung.

2.2.1 Funktionsweise

Die Funktionsweise des Plugins ist eigentlich relativ simpel. Der Benutzer konfiguriert nebst diversen Filter-Parametern einen Warning- sowie einen Critical-Threshold. Anschliessend filtert das Plugin anhand der konfigurierten Parameter das lokale systemd Journal und zählt die übrig gebliebenen Log-Einträge. Liegt die Anzahl der übrig gebliebenen Log-Einträge unterhalb der konfigurierten Thresholds, so wird der Statuscode “OK” zurückgegeben, liegt die Anzahl hingegen über einem der Thresholds, so wird entsprechend der Statuscode “Warning” oder “Critical” zurückgegeben.

In gewissen Situationen kann es unter Umständen interessant sein, anstatt einen globalen Counter zu verwenden, die übrig gebliebenen Log-Einträge zu gruppieren und für jede Gruppe einen separaten Counter zu verwenden. Das Check-Journald Plugin ermöglicht dies durch die konfiguration einer Regular Expression mit entsprechender capturing Group. Die Verwendung von Regular Expressions mit mehr als einer capturing Group ist jedoch momentan nicht unterstützt. Für mehr Informationen bezüglich der Python Regular Expression Syntax und Limiten der Konfiguration siehe Kapitel 4 sowie die Python Regular Expression Dokumentation [2].

2.2.2 CLI-Argumente

Um das oben genannte Filterung sowie der Threshold für den Benutzer konfigurierbar zu gestalten, bietet das Check-Journald Plugin eine Vielzahl von Command-Line-Interface (CLI) Argumenten. Im Folgenden werden alle verfügbaren CLI-Argumente mit ihren Switches, Standardwerten und Funktionsweisen ausführlich erläutert.

Critical Threshold:
Der Critical Threshold legt die Anzahl Log-Einträge fest, die erreicht werden muss, damit das Plugin den Statuscode “Critical” zurückgibt.
Switches: -c <number>
--critical <number>

Warning Threshold:
Der Warning Threshold legt die Anzahl Log-Einträge fest, die erreicht werden muss, damit das Plugin den Statuscode “Warning” zurückgibt. Dieser Wert sollte stets kleiner sein, als der Critical Threshold.
Switches: -w <number>
--warning <number>

Matches:
Übergibt die nachfolgende Liste von Matches, die dazu verwendet wird, das systemd Journal zu filtern. Matches müssen im folgenden Format definiert werden: FIELD=value, wobei FIELD einem systemd-journal-field entspricht. Für eine Liste aller üblichen Journald-Fields, siehe systemd.journal-fields [3]. Beispiel eines Matches: _SYSLOG_IDENTIFIER=sshd
Switches: -m <list of matches>
--matches <list of matches>

Regex:
Übergibt die nachfolgende Regular Expression, die dazu verwendet wird, das systemd Journal zu filtern. Anders als Matches mit dem Journald-Field “MESSAGE”, die nur auf exaktes übereinstimmen der Log-Nachrichten prüfen, kann mit Hilfe der Regular Expression auch auf ein teilweises Übereinstimmen geprüft werden. Ausserdem können die übrig gebliebenen Log-Einträge mit Hilfe einer capturing Group gruppiert werden. Für mehr Informationen bzgl. Regular Expressions und capturing Groups, siehe Kapitel 2.2.1, 4 sowie [3].
Switches: -r <regular expression>
--regex <regular expression>

Period:
Legt die Periode fest, deren Log-Einträge in Betracht gezogen werden. Eine Periode beginnt immer in der Vergangenheit und endet mit dem aktuellen Zeitpunkt. Perioden werden folgendermassen angegeben: 1-99m/h/d, wobei m für Minuten, h für Stunden, und d für Tage steht. Beispiel: 1h - Log-Einträge der letzten Stunde, 30m - Log-Einträge der letzten 30 minuten, etc.
Switches: --period <period>

Path:
Sollten die systemd Journals nicht im Standardverzeichnis gespeichert sein, oder will man mit systemd-journal-remote übermittelte Journals überwachen, so kann mit Hilfe der dieser Option das Verzeichnis der Journal-Files angegeben werden.
Switches: --path </path/to/journal/folder>

Verbose:
Aktiviert die ausführliche Konsolen-Ausgabe des Plugins. Listet den Timestamp sowie die Nachricht aller nach dem Filtering übrig gebliebenen Log-Einträge.
Switches: -v

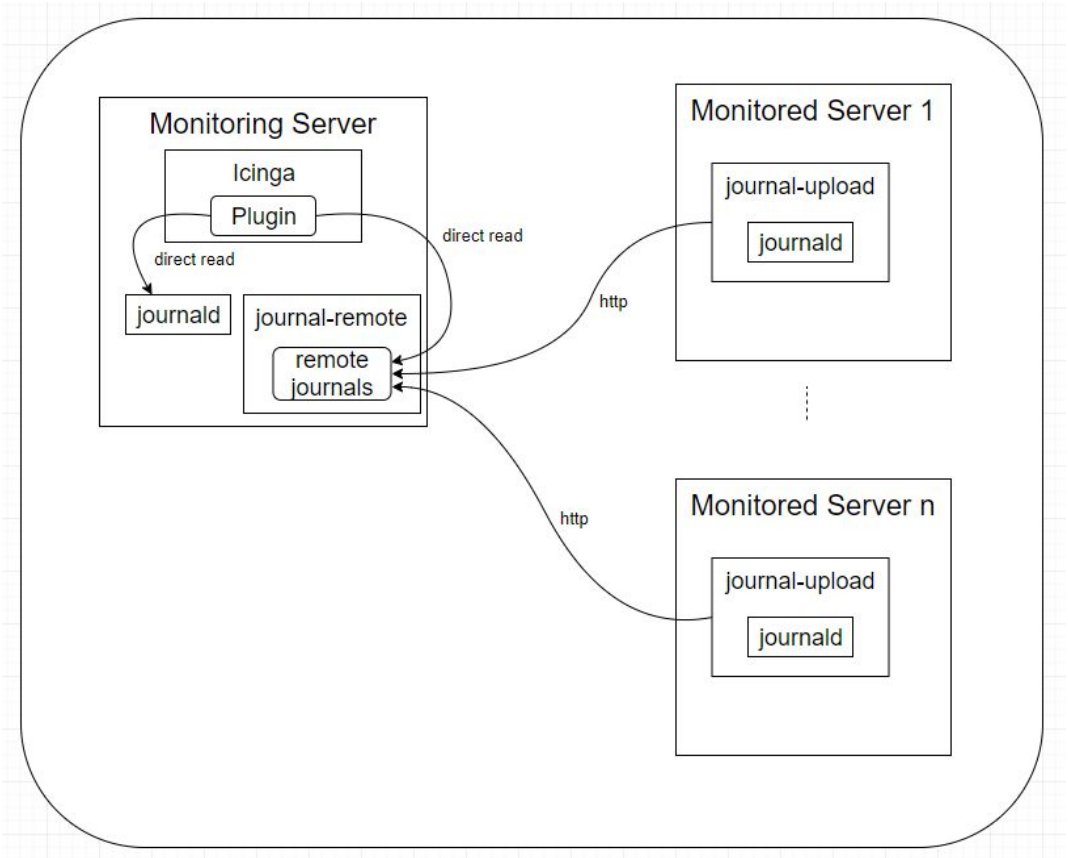
Help:
Gibt die Dokumentation der CLI-Argumente in der Konsole aus.
Switches -h

2.2.3 Remote Journal Access

Da das Plugin Check-Journald nur auf das lokale Journal zugreifen kann, werden im folgenden zwei Varianten beschrieben, wie remote Journals überwacht werden können.

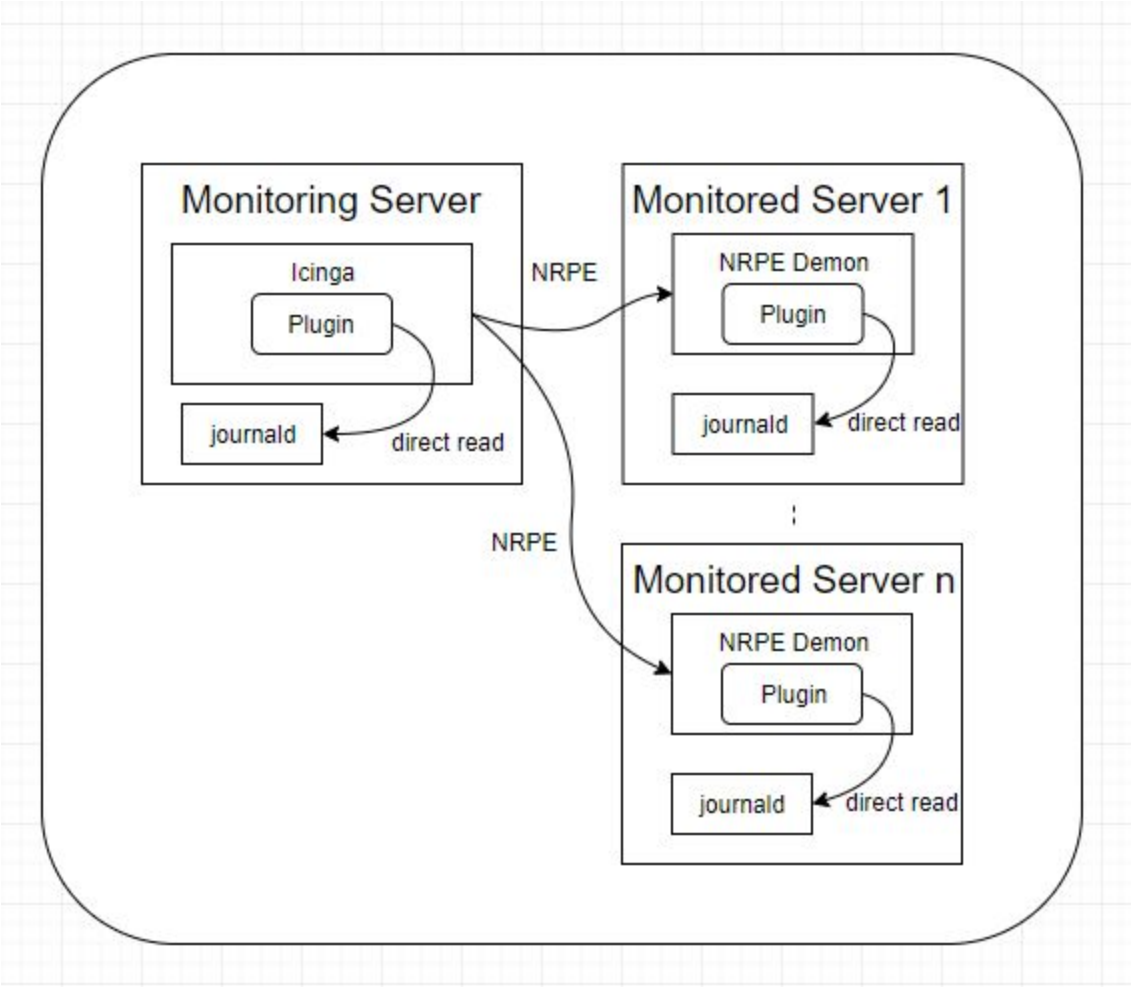
Journal-Upload / Journal-Remote:
Variante 1 verwendet die systemd Komponenten journal-upload (client seitig) sowie journal-remote (server seitig). Diese bieten die Möglichkeit, Journal-Files von mehreren Clients an den Icinga Server mit dem Check-Journald Plugin zu übermitteln. Der Server speichert die Journals lokal und überwacht sie mit Hilfe des Plugins (gegebenenfalls muss in Icinga der entsprechende Pfad konfiguriert werden, an dem die Journals gespeichert werden).

Der grosse Nachteil dieser Variante liegt darin, dass mit journal-upload bzw. journal-remote jeweils die gesamten Logfiles an den Server übermittelt werden. Dies führt einerseits dazu, dass serverseitig entsprechend mehr Speicher für das Aufbewahren der Logfiles benötigt wird, andererseits wird dadurch auch entsprechend viel Netzwerk Traffic generiert.



Nagios Remote Plugin Execution (NRPE):
Variante 2 verwendet den Service (Daemon) NRPE. Dieser erlaubt es Icinga2 einen Remote-Plugin-Call durchzuführen, d.h. ein Plugin auf einer Remote-Maschine auszuführen, welches dort lokal installiert ist. Installiert man NRPE zusammen mit dem Check-Journald Plugin auf der Remote-Maschine, so kann das Plugin dort das lokale Journal überwachen und übermittelt lediglich die Konsolen-Ausgabe sowie den Exit-Code des Plugin zurück an den Icinga Server.

Der grosse Vorteil dieser Variante liegt darin, dass lediglich die CLI-Argumente sowie die Konsolen-Ausgabe und der Exit-Code des Plugins übermittelt werden müssen. Die ganzen Logfiles bleiben lokal gespeichert wodurch kein nennenswerter Netzwerk-Traffic generiert wird.



3 Installation und Konfiguration

3.1 Voraussetzungen

Für die Verwendung des Check-Journald Plugins auf einem Linux-System muss nebst einer aktuellen Python3 (v3.6.7) und Icinga2 Installation auch eine aktuelle Version des Python Moduls “python-systemd” (v234) installiert sein. Während die aktuellen Versionen von Ubuntu bereits mit einer vorinstallierten Python3 Version ausgeliefert werden, muss das “python-systemd” Modul in jedem Fall separat installiert werden. Dazu kann der folgende Terminal Befehl verwendet werden:

```
sudo apt-get install python-systemd python3-systemd
```

Für weitere Informationen bzgl. der Installation von Icinga2, siehe [4].

Zusätzlich muss der ausführende Benutzer (üblicherweise “nagios”) zum Zugriff auf das systemd Journal berechtigt werden. Dazu muss der Benutzer der System-Gruppe “systemd-journal” hinzugefügt werden. Mit folgendem Terminal Befehl lässt sich dies bewerkstelligen:

```
sudo usermod -aG systemd-journal nagios
```

Da das Plugin von einem Daemon/Service ausgeführt wird, kann es sein, dass an der Stelle ein Neustart des Systems nötig ist, damit die neuen Berechtigungen korrekt geladen werden.

3.1.1 Remote Journal Access

Sollen zusätzlich zum lokalen Journal auch remote Journals überwacht werden, so gibt es je nach verwendeter remote Access Variante zusätzliche Voraussetzungen.

Journal-Remote/Journal-Upload:
Will man journal-remote und journal-upload verwenden um die Logfiles zu übermitteln, so kann es sein dass man diese Komponenten erst noch installieren muss. Dazu kann man den folgenden Terminal Befehl verwenden.

```
sudo apt-get install systemd-journal-remote
```

Für mehr Informationen bzgl. der Konfiguration von journal-remote und journal-update, siehe

```
man systemd-journal-remote
man systemd-journal-upload
```

NRPE:
Will man stattdessen NRPE verwenden, so muss man dies selbstverständlich auf der remote Maschine installieren. Dabei gibt es allerdings einige Dinge zu beachten. NRPE lässt sich auf zwei verschiedene Arten installieren: als Daemon oder via inetd bzw. xinetd, wobei die Installation als Daemon für den Gebrauch des Check-Journald Plugins bevorzugt wird.

Wir empfehlen für die Installation von NRPE das folgende Vorgehen:

```
// Voraussetzungen für die NRPE Installation
sudo apt-get install make
sudo apt-get install gcc   // Oder anderer C-Compiler
sudo apt-get install libssl-dev

// Download Folder erstellen
mkdir download
cd download

// Nagios-Plugins installation
wget http://nagios-plugins.org/download/nagios-
plugins-2.2.1.tar.gz
tar xzf nagios-plugins-2.2.1.tar.gz
cd nagios-plugins-2.2.1
./configure
sudo make
sudo make install
cd ..

//NRPE Installation
wget https://github.com/NagiosEnterprises/nrpe/
releases/download/nrpe-3.2.1/nrpe-3.2.1.tar.gz
tar xzf nrpe-3.2.1.tar.gz
cd nrpe-3.2.1
./configure --enable-command-args          //Notiz: 1
sudo make all
sudo make install-groups-users             //Notiz: 2
sudo make install
sudo make install-config
sudo make install-init
systemctl enable nrpe && systemctl start nrpe
```

Notiz 1: Das Flag “enable-command-args” ist zwingend nötig, wenn man die CLI-Argumente des Plugins vom Icinga2 Server aus konfigurieren und übermitteln will.
Notiz 2: Der dadurch neu angelegte Benutzer muss zusätzlich noch in die Gruppe “systemd-journal” eingetragen werden, damit er Zugriff auf das System Journal erhält.

Wird NRPE hingegen über inetd bzw. xinetd verwendet, so muss man das Plugin mit root Rechten ausführen, damit das Journal ordnungsgemäss ausgelesen werden kann (Siehe Kapitel 4). Dazu muss zunächst der entsprechende Benutzer in “sudoers” File eingetragen werden.

```
sudo visudo
```

Im Sudoers-File muss anschliessend die folgende Zeile ergänzt werden:

```
nagios ALL=NOPASSWD:
<PluginDir>/check_journald/check_journald.py
```

Ausserdem muss die Check-Command Definition im NRPE Konfigurationsfile mit dem “sudo” Präfix versehen werden (Siehe Kapitel 3.2)

Nach der Installation von NRPE muss zusätzlich noch die IP Adresse des Icinga Servers in das NRPE Konfigurations File eintragen und will man wie in Notiz 1 erwähnt die CLI-Argumente mit übermitteln, so muss auch die entsprechende Option aktiviert werden. Dazu ändert man im folgenden File die folgenden Einträge:

```
sudo nano /usr/local/nagios/etc/nrpe.cfg

allowed_hosts=XXX.XXX.XXX.XXX
dont_blame_nrpe=1    //Potentielles Sicherheitsrisiko!
```

Für weiter Informationen bzgl. der Installation von NRPE, siehe NRPE Dokumentation [6].

3.2 Plugin Installation

Die Installation des Plugins ist relativ simpel. Man kopiert den check_journald Ordner in das entsprechende Plugin-Verzeichnis. Bei Icinga ist dies standardmässig:

```
/usr/lib/nagios/plugins/
```

Bei NRPE ist dies:

```
/usr/local/nagios/libexec/
```

Anschliessend konfiguriert man einen entsprechenden Check-Command.

3.3 Icinga2 Beispiel Konfiguration

Beispiel eines CheckCommand Objekt, welches via NRPE “check_journald” aufruft und die Argumente vom Service weiterreicht.

```
/* Command config: /etc/icinga2/conf.d/commands.conf */
object CheckCommand "nrpe_check_journald" {
   command = [ PluginDir + "/check_nrpe" ]

   arguments += {
       "-H" = {   /* host with nrpe runing */
           required = true
           value = "$address$"   /* pass on address variable */
           order = 1
       }
       "-c" = {  /* name of the command on the remote host */
           required = true
           value = "check_journald"
           order = 2
       }
       "-a" = {   /* arguments pass on from variable */
           value = "$check_journald_arguments$"
           order = 3   /* arguments need to be last */
       }
   }
}
```

Beispiel eines minimalen Host Objekt für dieses Beispiel.

```
/* Host config: /etc/icinga2/conf.d/hosts.conf */
object Host "RGServer" {
   import "generic-host"

   address = "91.214.169.165"
}
```

Icinga2 Service Objekt mit dem Beispiel Command und Beispiel Parameter.

```
/* Service config: /etc/icinga2/conf.d/services.conf */
apply Service "Failed SSH Logins" {
   import "generic-service"

   check_command = "nrpe_check_journald"

   vars.check_journald_arguments = "--warnig 5 --critical 10
--matches SYSLOG_IDENTIFIER=sshd --regex \"Failed assword for
.* from (.+) port\""

   assign where host.name == "RGServer"   /* assign to host */
}
```

3.4 NRPE Beispiel Konfiguration

Beispiel einer NRPE Check-Command Definition, welche CLI-Argumente von Icinga an das Plugin weiterleitet.

```
command[check_journald]=
     <PluginDir>/check_journald/check_journald.py $ARG1$
```

Oder für den Fall dass NRPE unter inetd läuft mit dem entsprechenden “sudo” Präfix:

```
command[check_journald]= sudo
     <PluginDir>/check_journald/check_journald.py $ARG1$
```

4 Rückblick: Probleme

Während der Entwicklung des Plugins gab es zwei nennenswerte Probleme, die nach wie vor noch nicht auf zufriedenstellende Art und Weise behoben werden konnten.

Journal Berechtigungen:
Das erste Problem hängt mit den systemd Journal Berechtigungen bzw. der Systemgruppe “systemd-journal” zusammen. Fügt man nämlich den entsprechenden Benutzer der “systemd-journal” Gruppe hinzu, so funktioniert das Plugin unter Icinga2 und NRPE als Daemon installiert einwandfrei. Läuft NRPE hingegen unter inetd oder xinetd, so werden die Berechtigungen offensichtlich nicht richtig gelesen, denn obwohl xinetd unter dem “root” Benutzer läuft und das Plugin unter dem entsprechend konfigurierten und berechtigten “nagios” Benutzer, so kann das Journal trotzdem nicht gelesen werden.

Regular Expression Syntax im Check-Command Objekt:
Das zweite Problem hängt mit der Python Regular Expression Syntax zusammen. Diese verwendet diverse Sonderzeichen, welche in einem Check-Command Objekt entsprechend escaped werden müssen. Dies ist nicht nur unschön und umständlich für den Benutzer, sondern teilweise sogar so Fehlerhaft, dass manche Check-Command Objekte schlichtweg nicht mehr akzeptiert werden, wenn entsprechende Regex Syntax Elemente verwendet werden. Dies schränkt den Nutzen der Regular Expression natürlich entsprechend stark ein, vor allem im Kontext des gruppierens von Log-Einträgen.

5 Ausblick

5.1 Weiterentwicklung

In der aktuellen Version des Plugins werden immer nur diejenigen Log-Einträge in Betracht gezogen, die in einer bestimmten (konfigurierbaren) Zeitperiode vor dem aktuellen Zeitpunkt liegen. Als alternative dazu könnte man auch eine sogenannte “Cookie” basierte Version implementieren. Dabei würde der zuletzt gelesene Log-Eintrag in einem Cookie (File) gespeichert und beim nächsten Aufruf des Plugins würden nur diejenigen Einträge gelesen, die neuer sind als der Eintrag im Cookie.

Ausserdem bietet das Python Modul python-systemd diverse weitere Filtermöglichkeiten, die man dem Benutzer über entsprechende CLI-Argumente zur Verfügung stellen könnte.

5.2 Bachelor Thesis

Unser Vorschlag für die Bachelor Thesis wäre es, ein journald basiertes Security Monitoring/ Intrusion Detection für Icinga2 zu implementieren. Als Basis dafür könnte das im Rahmen dieses Projektes entwickelte Plugin dienen. Dazu könnte sowohl das Plugin weiterentwickelt werden, sowie geeignete Konfigurationen für das Plugin herausgearbeitet werden. Damit könnten beispielsweise die folgenden drei Szenarien abgedeckt werden:

- Unerwartete Veränderungen an lokalen Benutzern und Gruppenzugehörigkeiten
- DNS Lookups von potenziell bösartigen Domains/IP-Adressen (bspw. Unter Verwendung von Spamhouse Domain Block Lists etc.)
- Unerwartetes Öffnen von Firewall Ports, sowie unerwartetes Starten und/oder Beenden von Services/Prozessen

6 Anhang

6.1 Weiterführende Links

[1] Monitoring Plugins Development Guidelines:
https://www.monitoring-plugins.org/doc/guidelines.html#AEN33

[2] Python Regular Expressions:
https://docs.python.org/3/library/re.html

[3] systemd.journal-fields:
https://www.freedesktop.org/software/systemd/man/systemd.journal-fields.html#

[4] Icinga2 Installation:
https://icinga.com/docs/icinga2/latest/doc/02-getting-started/

[5] Python Modul: python-systemd
https://www.freedesktop.org/software/systemd/python-systemd/journal.html

[6] NRPE Dokumentation:
https://assets.nagios.com/downloads/nagioscore/docs/nrpe/NRPE.pdf

6.2 Zeitplan


Schedule
Woche 1 27/02/2019
Woche 2 06/03/2019
Woche 3 13/03/2019
Woche 4 20/03/2019
Woche 5 27/03/2019
Woche 6 03/04/2019
Woche 7 10/04/2019
Woche 8 17/04/2019
Woche 9 24/04/2019
Woche 10 01/05/2019
Woche 11 08/05/2019
Woche 12 15/05/2019
Woche 13 22/05/2019
Woche 14 29/05/2019
Woche 15 05/06/2019
Woche 16 12/06/2019
Projektplanung Soll
Recherche: systemd, icinga API, syslog & usw. Soll
Evaluation/Konzeption Soll
Entwicklung/Implementation Soll
Testing Soll
Präsentation Soll
Dokumentation Soll
1
2
3
4
5
Sollzeit nach Zeitplan
1 Abschluss Projektplanung
2 Abschluss Konzeption
3 Abschluss Entwicklung/Implementatio
4 Präsentation
5 Abgabe

6.3 Programmcode

https://github.com/MrTinnysis/project2

6.3.1 check_journald.py

```
#!/usr/bin/env python3

import re, sys, argparse, os
from JournalReader import JournalReader

# define period
def period(string):
    if not re.search("^\d{1,2}[dhm]$", string):
        msg = "%r is not a valid period" % string
        raise argparse.ArgumentTypeError(msg)
    return string

def printPerformanceData(label, ctr, warn, crit):
      data = ["%s=%s" % (label, str(ctr)), str(warn), str(crit)]
      print("|" + ";".join(data))

def main():
      # monitoring plugin return codes
      OK = 0
      WARNING = 1
      CRITICAL = 2
      UNKNOWN = 3

      # defaults
      returnCode = UNKNOWN
      # parse arguments
      argumentParser = argparse.ArgumentParser()

      argumentParser.add_argument(
            '-v', '--verbose', nargs="?", const=True, default=False,
            help='verbose output')
      argumentParser.add_argument(
            '-w', '--warning', metavar='NUMBER', type=int, default=1,
            help='return warning if count of found logs is outside RANGE')
      argumentParser.add_argument(
            '-c', '--critical', metavar='NUMBER', type=int, default=2,
            help='return critical if count of found logs is outside RANGE')
      argumentParser.add_argument(
            '--path',
            help='path to journal log folder')
      argumentParser.add_argument(
```

```
                '-m', '--matches', nargs='+',
                help='matches for logparse')
        argumentParser.add_argument(
                '--period', metavar='NUMBER', default='1h', type=period,
                help='check log of last period (default: "1h", format 1-99
m/h/d)')
        argumentParser.add_argument(
                '-r', '--regex',
                help='Regular expression to match message content'
        )

        arguments = argumentParser.parse_args()

        print(arguments)

        #setup journal reader
        journal = JournalReader(arguments.path)

        if arguments.matches:
                journal.add_matches(arguments.matches)


        journal.set_timeframe(arguments.period)

        if arguments.regex:
                regex = re.compile(arguments.regex)
                # filter journal by regex
                entries = [entry for entry in journal if
                                        regex.search(entry["MESSAGE"])]
                journal.close()

                if regex.groups == 1:
                        ctr = {}
                else:
                        ctr = 0
        else:
                entries = [entry for entry in journal]
                journal.close()
                ctr = 0


        #count journal entries
        for entry in entries:
                if arguments.verbose:
                        print(str(entry["__REALTIME_TIMESTAMP"]) +
                                        ": " + entry["MESSAGE"], end="\n")

                if type(ctr) is dict:
                        match = regex.search(entry["MESSAGE"])
                        ctr.setdefault(match.group(1), 0)
```

```
                        ctr[match.group(1)] += 1
                else:
                        ctr += 1


        returnCode = OK

        if type(ctr) is dict:
                for key, val in ctr.items():
                        if val in range(arguments.warning, arguments.critical-1):
                                returnCode = max(returnCode, WARNING)

                        if val >= arguments.critical:
                                returnCode = max(returnCode, CRITICAL)

                        printPerformanceData(key, val, arguments.warning,
                                        arguments.critical)
        else:
                if ctr in range(arguments.warning, arguments.critical-1):
                        returnCode = WARNING


                if ctr >= arguments.critical:
                        returnCode = CRITICAL

                printPerformanceData("count", ctr, arguments.warning,
                                        arguments.critical)


        sys.exit(returnCode)

if __name__=="__main__":
        main()
```

6.3.2 JournalReader.py

```
#!/bin/python

import re
from datetime import datetime, timedelta
from systemd import journal

class JournalReader(journal.Reader):

        def __init__(self, path=None):
                super(JournalReader, self).__init__(path=path)

        def add_matches(self, matches):
                for match in matches:
```

```
                    self.add_match(match)

        def set_timeframe(self, timeframe):
                start_time = datetime.now()
                match = re.match('(\d{1,2})([dhm])', timeframe)

                if not match:
                        raise InvalidTimeframeException()
                else:
                        quantity = max(int(match.group(1)), 1)
                        identifier = match.group(2)

                        start_time -= self._timeDelta(quantity, identifier)


                        print("Accessing log entries after: %s" % (start_time),
                                                end="\n")
                        self.seek_realtime(start_time)


        def _timeDelta(self, quantity, identifier):
                if identifier == "d":
                        return timedelta(days=quantity);
                elif identifier == "h":
                        return timedelta(hours=quantity);
                elif identifier == "m":
                        return timedelta(minutes=quantity);
                else:
                        raise InvalidTimeframeException()

class InvalidTimeframeException(Exception):
        pass
```