Universität Rostock
Traditio et Innovatio

Service- and Security Monitoring

Konsolidierung, Korrelierung und Visualisierung IT-Sicherheitskritischer Ereignisse

Martin Steinbach
Universität Rostock, Institut für Informatik

Seminar Sommersemester 2018
Aufbereitung und Auswertung komplexer Daten

Struktur

Struktur

Struktur

Struktur

Struktur

Monitoring!

Servicemonitoring = Securitymonitoring?

Ziele der Informationssicherheit

- Vertraulichkeit
- **Verfügbarkeit**
- **Verbindlichkeit**
- **Integrität**

Betrachtungen

- Zeitliche Perspektive
- Schweregrad (severity)
- Quelle
- Ereigniskorrelation

Ziele der Informationssicherheit

- Vertraulichkeit
- **Verfügbarkeit**
- **Verbindlichkeit**
- **Integrität**

Betrachtungen

- Zeitliche Perspektive
- Schweregrad (severity)
- Quelle
- Ereigniskorrelation

Ziele der Informationssicherheit

- Vertraulichkeit
- **Verfügbarkeit**
- **Verbindlichkeit**
- **Integrität**

Betrachtungen

- Zeitliche Perspektive
- Schweregrad (severity)
- Quelle
- Ereigniskorrelation

Warum Überwachung?

Schaden identifizieren und vermeiden

Zivil

- 90% aller Firmen: Opfer von Cyberattacken
- 80% derer mit finanziellen Einbußen
- Diebstahl geistigen Eigentums (zwischen 2011 und 2015 Verdopplung)

Hoheitlich / Kritische Infrastrukturen / Cyberabwehr / Militär

- Nationales IT-Lagezentrum
- NCAZ
- CIR (Bw)

Servicemonitoring = Securitymonitoring?

Ja!?

Nur ein nicht manipulierter Dienst, der erwiesenermaßen seine Aufgaben erfüllt, kann die Ziele der IT-Sicherheit einhalten.

Beweis durch Überwachung

- Unerwartetes Verhalten
- Erreichbarkeit
- Angriffserkennung
- Nachvollziebarkeit

Servicemonitoring = Securitymonitoring?

Ja!?

Nur ein nicht manipulierter Dienst, der erwiesenermaßen seine Aufgaben erfüllt, kann die Ziele der IT-Sicherheit einhalten.

Beweis durch Überwachung

- Unerwartetes Verhalten
- Erreichbarkeit
- Angriffserkennung
- Nachvollziebarkeit

Servicemonitoring = Securitymonitoring?

Ja!?

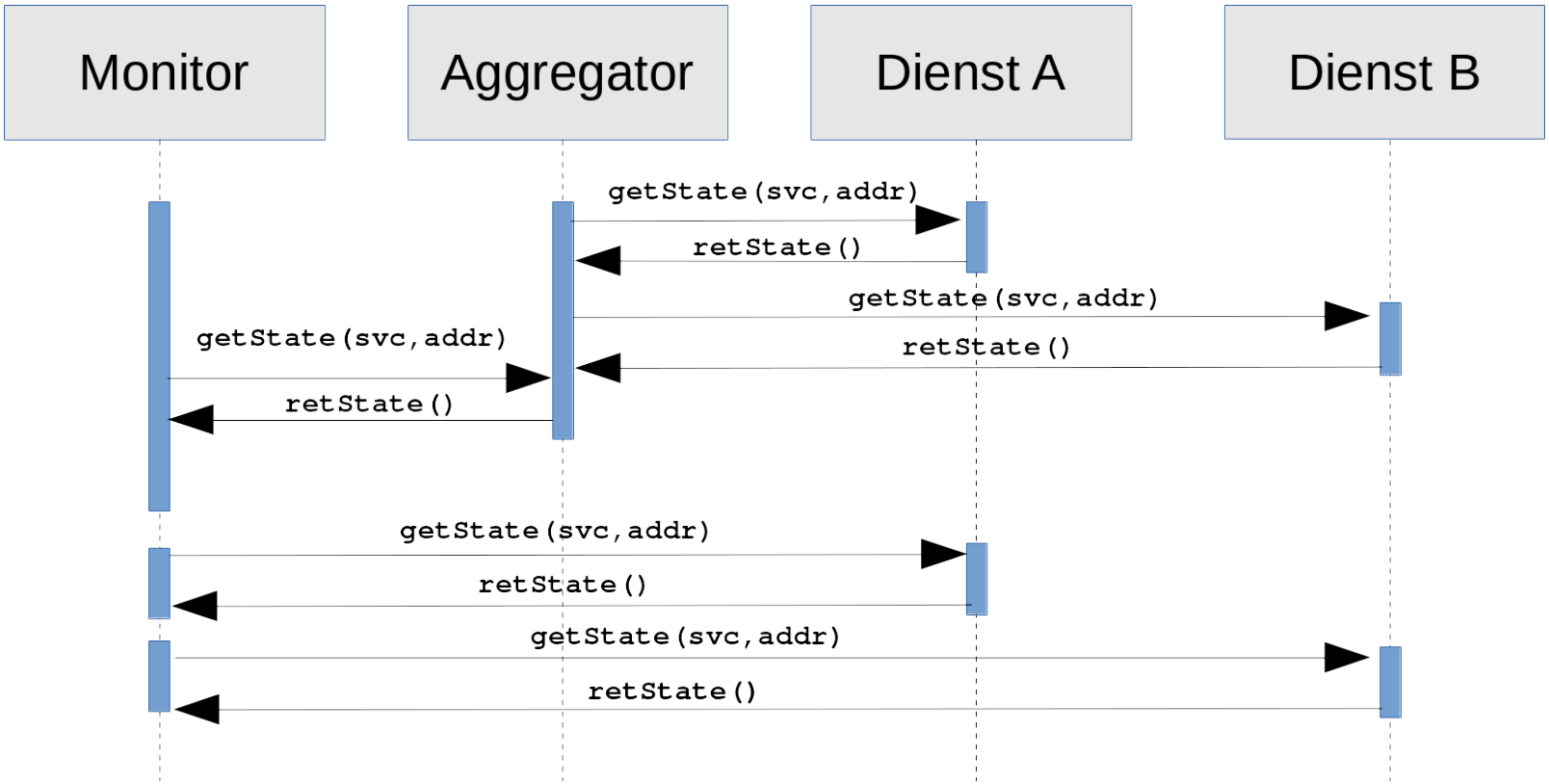
Nur ein nicht manipulierter Dienst, der erwiesenermaßen seine Aufgaben erfüllt, kann die Ziele der IT-Sicherheit einhalten.

Beweis durch Überwachung

- Unerwartetes Verhalten
- Erreichbarkeit
- Angriffserkennung
- Nachvollziebarkeit

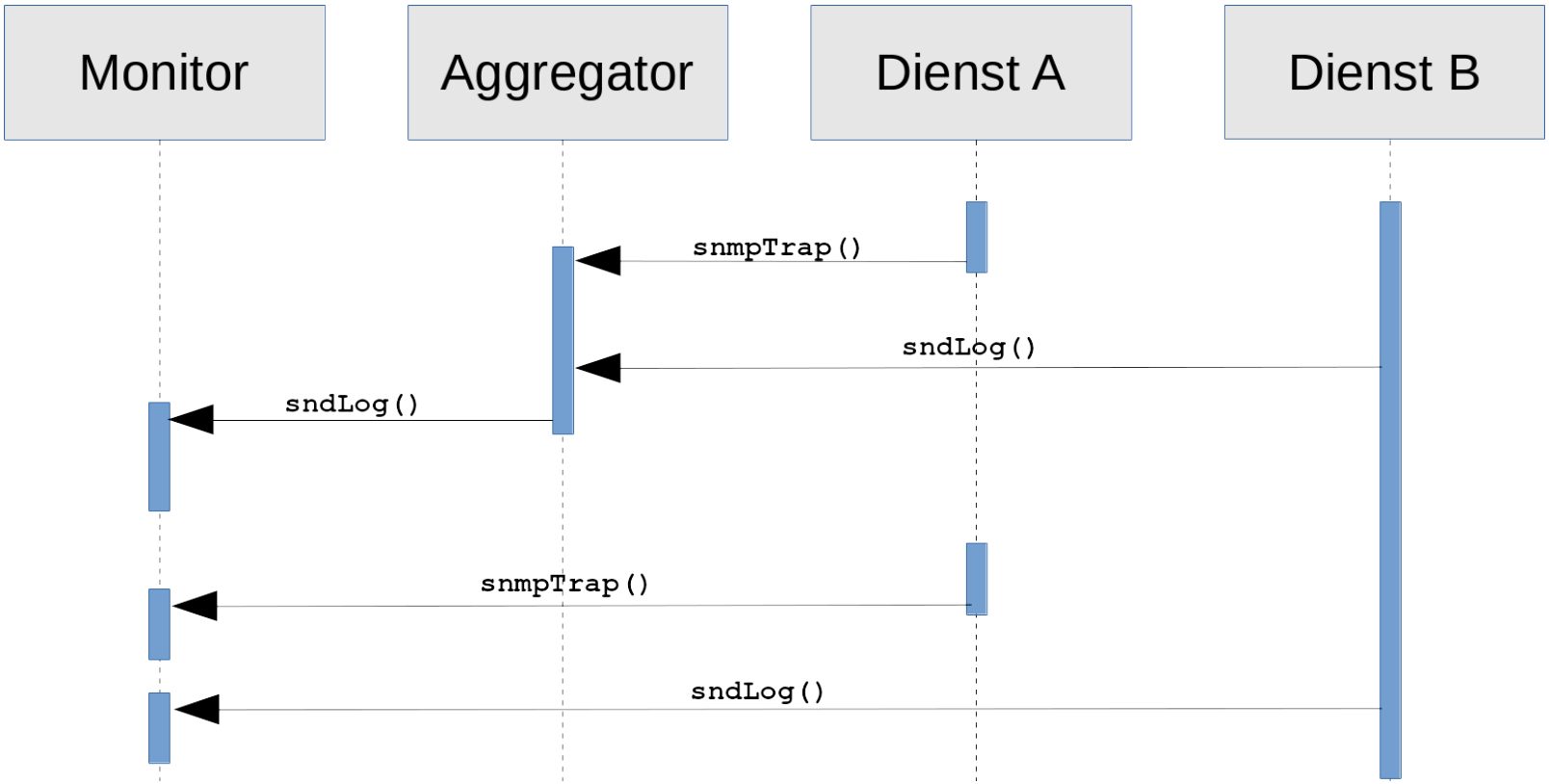
Aktive Überwachung

Beispiel Sequenzdiagramm



Passive Überwachung

Beispiel Sequenzdiagramm



Aktive Überwachung

Was kann überwacht werden?

Jede Entität, deren Status deutlich zueinander abgrenzbar sind.

- Betriebssystemabhängig
  - Betriebssystemparameter
    - Auslastung
    - Speicher
    - Prozesse
    - Datendurchsatz
  - Updates
  - Sicherheitsauditierung
- Betriebssystemunabhängig
  - Netzwerkdienste
    - L3: ICMP{4,6}
    - L4: TCP, UDP - basierend
    - L4+: SNMP
  - Sensoren
  - Aktive Netzwerkkomponenten

Aktive Überwachung

Was kann überwacht werden?

Jede Entität, deren Status deutlich zueinander abgrenzbar sind.

- Betriebssystemabhängig
  - Betriebssystemparameter
    - Auslastung
    - Speicher
    - Prozesse
    - Datendurchsatz
  - Updates
  - Sicherheitsauditierung
- Betriebssystemunabhängig
  - Netzwerkdienste
    - L3: ICMP{4,6}
    - L4: TCP, UDP - basierend
    - L4+: SNMP
  - Sensoren
  - Aktive Netzwerkkomponenten

Aktive Überwachung

Was kann überwacht werden?

Jede Entität, deren Status deutlich zueinander abgrenzbar sind.

- Betriebssystemabhängig
  - Betriebssystemparameter
    - Auslastung
    - Speicher
    - Prozesse
    - Datendurchsatz
  - Updates
  - Sicherheitsauditierung

- Betriebssystemunabhängig
  - Netzwerkdienste
    - L3: ICMP{4,6}
    - L4: TCP, UDP - basierend
    - L4+: SNMP
  - Sensoren
  - Aktive Netzwerkkomponenten

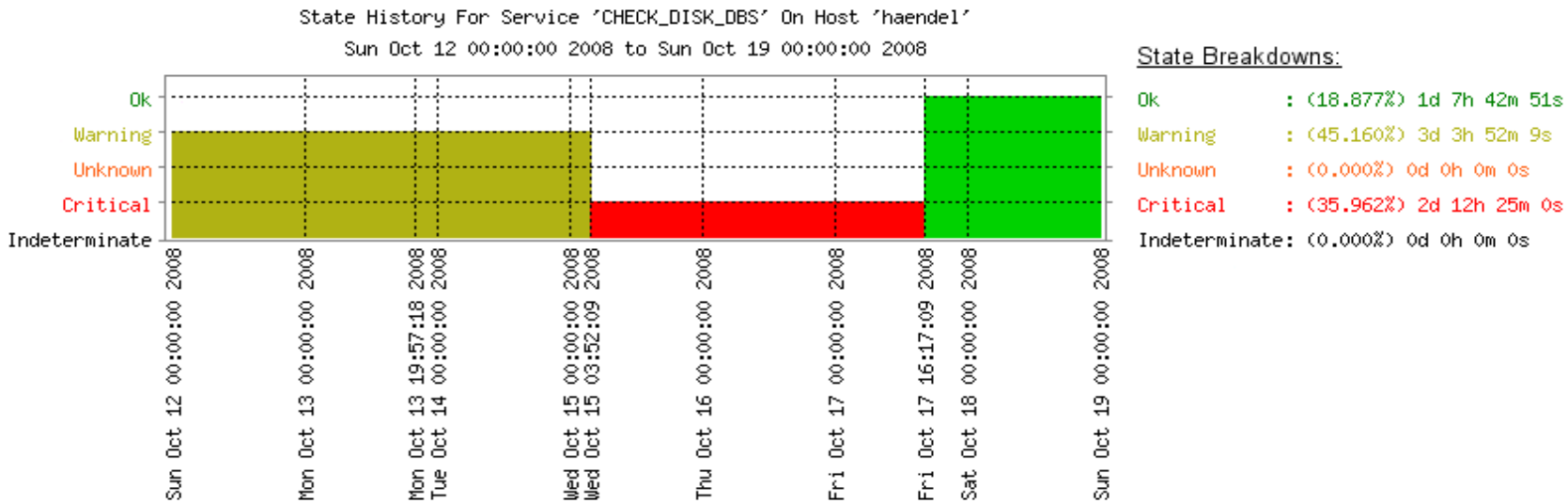
Aktive Überwachung

Was kann überwacht werden?

Jede Entität, deren Status deutlich zueinander abgrenzbar sind.

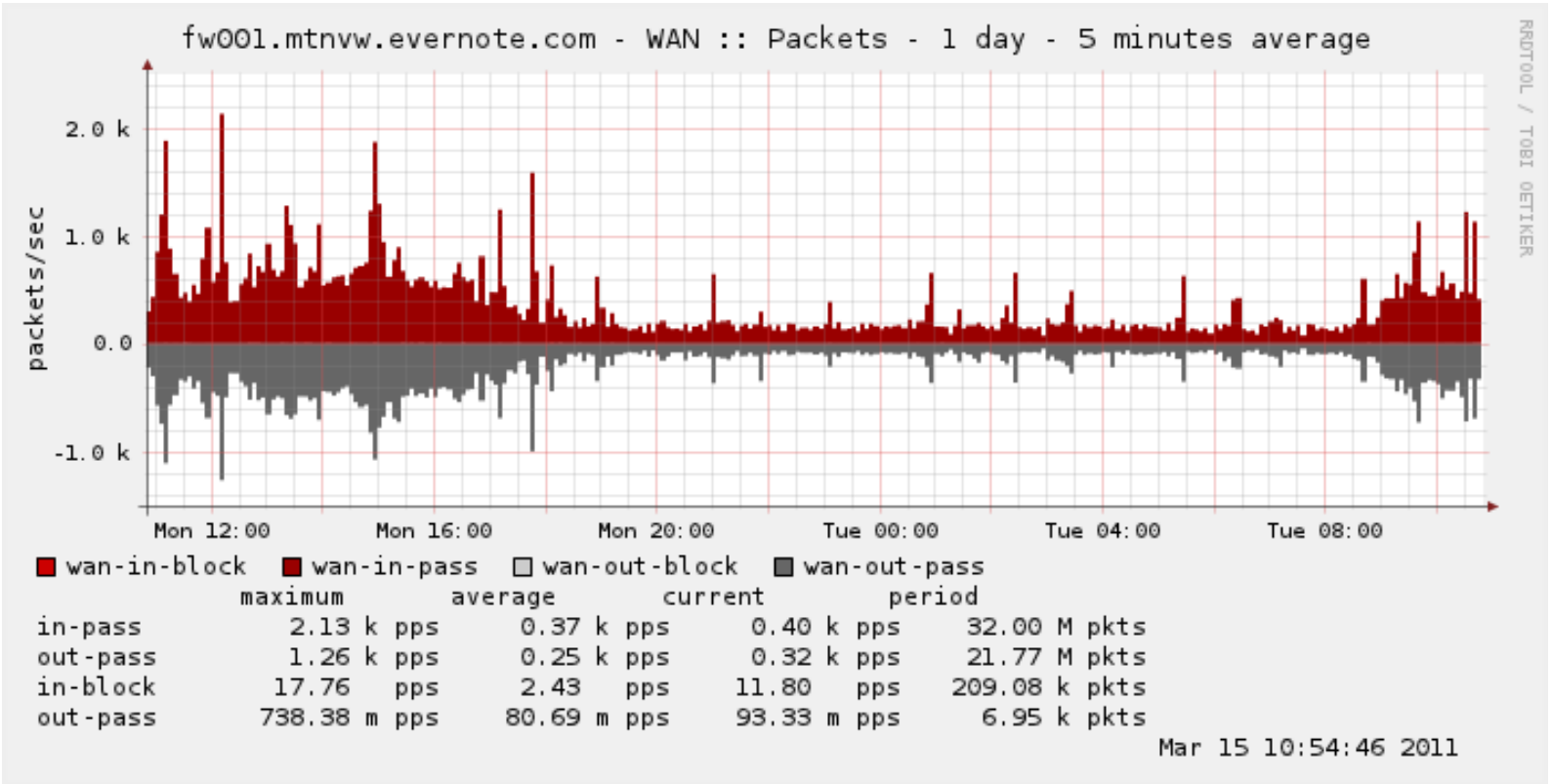
- Betriebssystemabhängig
  - Betriebssystemparameter
    - Auslastung
    - Speicher
    - Prozesse
    - Datendurchsatz
  - Updates
  - Sicherheitsauditierung

- Betriebssystemunabhängig
  - Netzwerkdienste
    - L3: ICMP{4,6}
    - L4: TCP, UDP - basierend
    - L4+: SNMP
  - Sensoren
  - Aktive Netzwerkkomponenten



Quelle: selbst erstellt
Erstellt mit: NagVis

Visualisierungen
packets/time



Quelle: https://redmine.pfsense.org/issues/1354
Erstellt mit: https://oss.oetiker.ch/rrdtool/

cloud - IaaS

- Stark steigende Systemanzahl (10K+)
- In wenigen Sekunden: virtuelles RZ
- Dynamisch wachsendes/sinkendes Logaufkommen
- Dynamische Kosten
- Proprietäre, inkompatible Monitoringsysteme [a]

[a] IETF-draft: *Syslog Extension for Cloud Using Syslog Structured Data*

Anforderungen Logkorrelation

- Manuell undurchführbar
- Skalierbar (n+1)
- Automatisch durchführbar
- Minimierung des Speicheraufwandes

cloud - IaaS

- Stark steigende Systemanzahl (10K+)
- In wenigen Sekunden: virtuelles RZ
- Dynamisch wachsendes/sinkendes Logaufkommen
- Dynamische Kosten
- Proprietäre, inkompatible Monitoringsysteme [a]

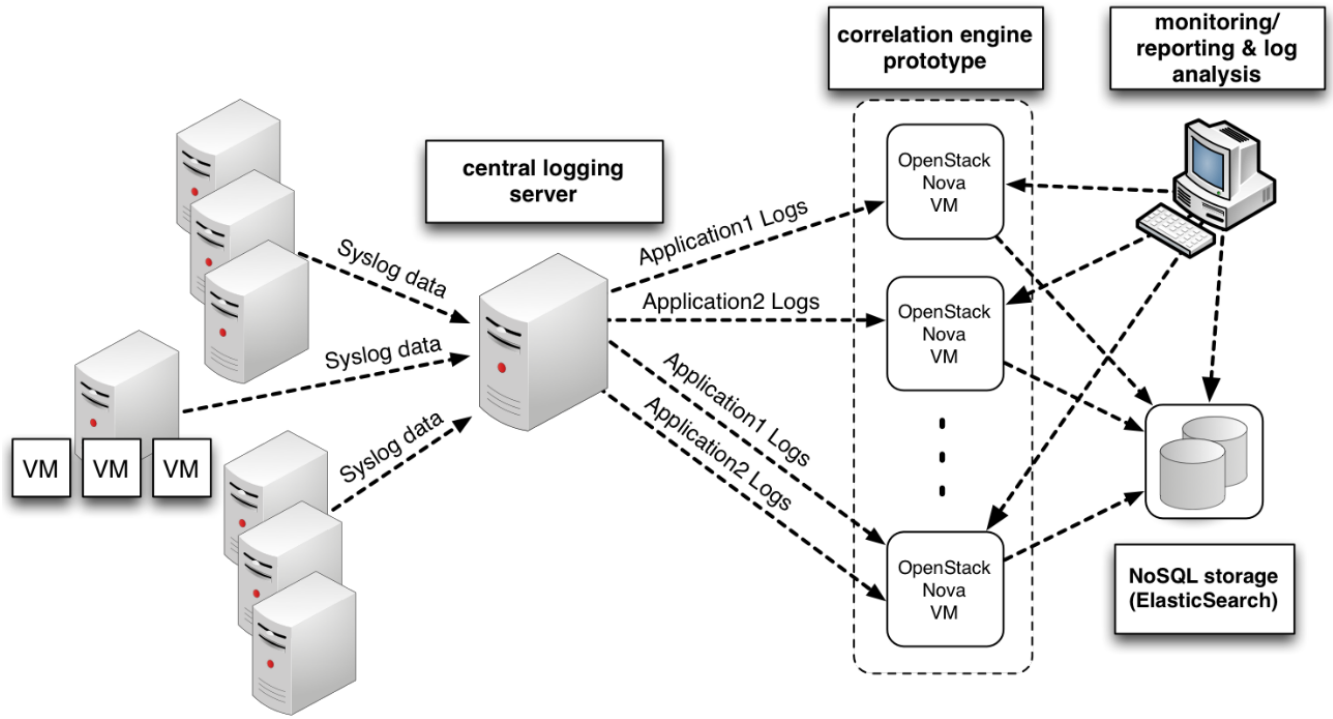
[a] IETF-draft: *Syslog Extension for Cloud Using Syslog Structured Data*

Anforderungen Logkorrelation

- Manuell undurchführbar
- Skalierbar (n+1)
- Automatisch durchführbar
- Minimierung des Speicheraufwandes

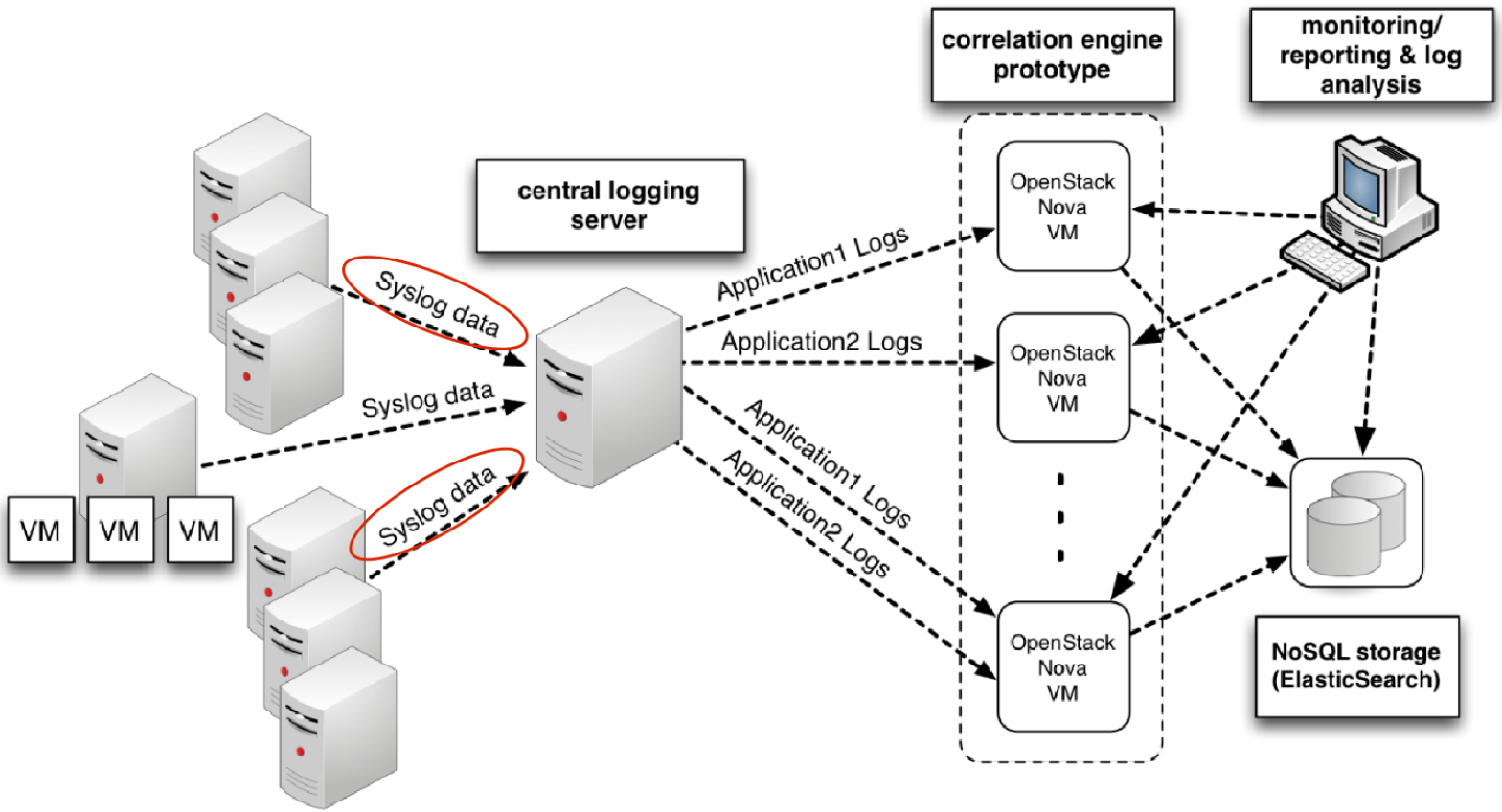
Prototyp JCorrelat

Schema



Quelle: D.Frisch ,C. Pape, S. Reissmann, and S. Rieger “Correlation and Consolidation of Distributed Logging Data in Enterprise Clouds” In International Journal on Advances in Internet Technology, vol 7 , 2013, pp. 39–51.

Syslog
Aufbau RFC 5424



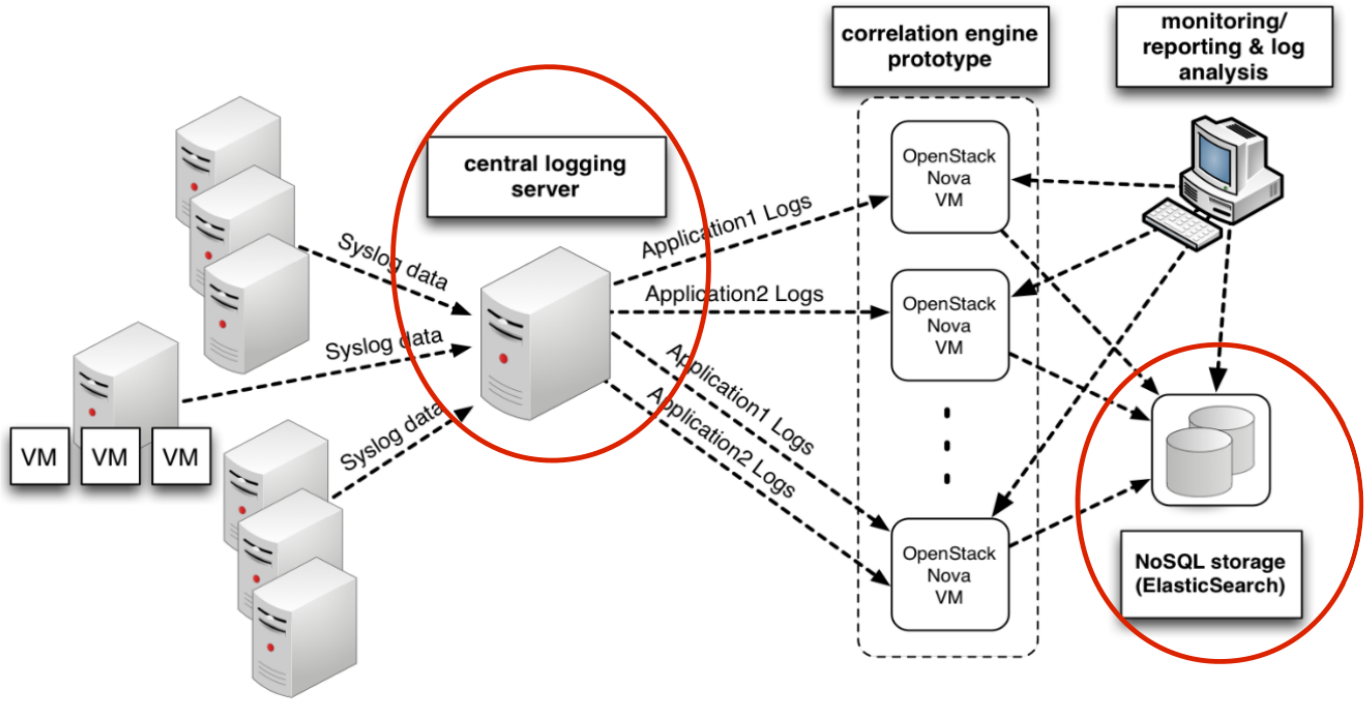
RFC5424 Implementiert durch syslog-ng und rsyslog

Feld	Inhalt	Beispiel
HEADER		
facility	$int \in \{0..23\}$	<**16**5> (local0)
severity	$int \in \{0..7\}$	<16**5**> (Notice)
timestamp	RFC3339	2003-10-11T22:14:15.003Z
hostname	string	mymachine.example.com
tag	string	evntslog
MSG		
MSGID	string	ID47
structured data	key=value	eventID="1011"
content	string	An application event log...

```
<165> 2003-10-11T22:14:15.003Z mymachine.example.com evntslog -
ID47 [exampleSDID@32473 iut="3" eventSource= "Application"
eventID="1011"] BOMAn application event log entry...
```

Syslog

Persistente Speicherung und Konsolidierung



Persistente Speicherung

Probleme

- Menge der anfallenden Daten
- Hohe Redundanz der Daten
- Durchsuchbarkeit Logdaten

Relationale Datenbanken

- Schema der Daten muss bekannt sein
- Schemaänderungen nur schwer möglich

NoSQL (Not only SQL)

- Kein festes Schema
- Wohlformatierte strukturierte Daten (key $\Rightarrow$ val)
- Sehr gut skalierbar

Persistente Speicherung

Probleme

- Menge der anfallenden Daten
- Hohe Redundanz der Daten
- Durchsuchbarkeit Logdaten

Relationale Datenbanken

- Schema der Daten muss bekannt sein
- Schemaänderungen nur schwer möglich

NoSQL (Not only SQL)

- Kein festes Schema
- Wohlformatierte strukturierte Daten (key $\Rightarrow$ val)
- Sehr gut skalierbar

Persistente Speicherung

Probleme

- Menge der anfallenden Daten
- Hohe Redundanz der Daten
- Durchsuchbarkeit Logdaten

Relationale Datenbanken

- Schema der Daten muss bekannt sein
- Schemaänderungen nur schwer möglich

NoSQL (Not only SQL)

- Kein festes Schema
- Wohlformatierte strukturierte Daten (key $\Rightarrow$ val)
- Sehr gut skalierbar

Persistente Speicherung

Probleme

- Menge der anfallenden Daten
- Hohe Redundanz der Daten
- Durchsuchbarkeit Logdaten

Relationale Datenbanken

- Schema der Daten muss bekannt sein
- Schemaänderungen nur schwer möglich

NoSQL (Not only SQL)

- Kein festes Schema
- Wohlformatierte strukturierte Daten (key $\Rightarrow$ val)
- Sehr gut skalierbar

Persistente Speicherung

Überblick über 3 NoSQL-Strategien

`key ⇒ value datastores`

- Hochperformant (key ⇒ BLOB)
- Einfache API (`insert, delete, lookup`)
- Keine Suche in BLOBs

`column-oriented datastores`

- Verwaltung in Zeilen und Spalten
- Skalierung: Auftrennung in *shards*

`document-based datastores`

- `document`: Menge an Objekten mit unterschiedlichen Attributen
- Skalierung: Aufteilung der Objekte
- Volltextsuche

key ⇒ value datastores

- Hochperformant (key ⇒ BLOB)
- Einfache API (`insert`, `delete`, `lookup`)
- Keine Suche in BLOBs

column-oriented datastores

- Verwaltung in Zeilen und Spalten
- Skalierung: Auftrennung in *shards*

document-based datastores

- `document`: Menge an Objekten mit unterschiedlichen Attributen
- Skalierung: Aufteilung der Objekte
- Volltextsuche

key ⇒ value datastores

- Hochperformant (key ⇒ BLOB)
- Einfache API (`insert, delete, lookup`)
- Keine Suche in BLOBs

column-oriented datastores

- Verwaltung in Zeilen und Spalten
- Skalierung: Auftrennung in *shards*

document-based datastores

- `document`: Menge an Objekten mit unterschiedlichen Attributen
- Skalierung: Aufteilung der Objekte
- Volltextsuche

key ⇒ value datastores

- Hochperformant (key ⇒ BLOB)
- Einfache API (`insert, delete, lookup`)
- Keine Suche in BLOBs

column-oriented datastores

- Verwaltung in Zeilen und Spalten
- Skalierung: Auftrennung in *shards*

document-based datastores

- `document`: Menge an Objekten mit unterschiedlichen Attributen
- Skalierung: Aufteilung der Objekte
- Volltextsuche

Ziel der Korrelation von Logdaten

1. **Reduktion des Datenaufkommens**
   - Konsolidierung
2. **Identifizierung wesentlicher Informationen**
   - Korrelation

Im Folgenden:
Beispielhafte Korrelation anhand einer SSH Brute-Force-Attacke.

Ziel: Den einen erfolgreichen Versuch dieser Attacke zu identifizieren.

Ziel der Korrelation von Logdaten

1. **Reduktion des Datenaufkommens**
   - Konsolidierung
2. **Identifizierung wesentlicher Informationen**
   - Korrelation

Im Folgenden:
Beispielhafte Korrelation anhand einer SSH Brute-Force-Attacke.

Ziel: Den einen erfolgreichen Versuch dieser Attacke zu identifizieren.

Ziel der Korrelation von Logdaten

1. **Reduktion des Datenaufkommens**
   - Konsolidierung
2. **Identifizierung wesentlicher Informationen**
   - Korrelation

Im Folgenden:
Beispielhafte Korrelation anhand einer SSH Brute-Force-Attacke.

Ziel: Den einen erfolgreichen Versuch dieser Attacke zu identifizieren.

Ziel der Korrelation von Logdaten

1. **Reduktion des Datenaufkommens**
   - Konsolidierung
2. **Identifizierung wesentlicher Informationen**
   - Korrelation

Im Folgenden:
Beispielhafte Korrelation anhand einer SSH Brute-Force-Attacke.

Ziel: Den einen erfolgreichen Versuch dieser Attacke zu identifizieren.

liblognorm - Regeln

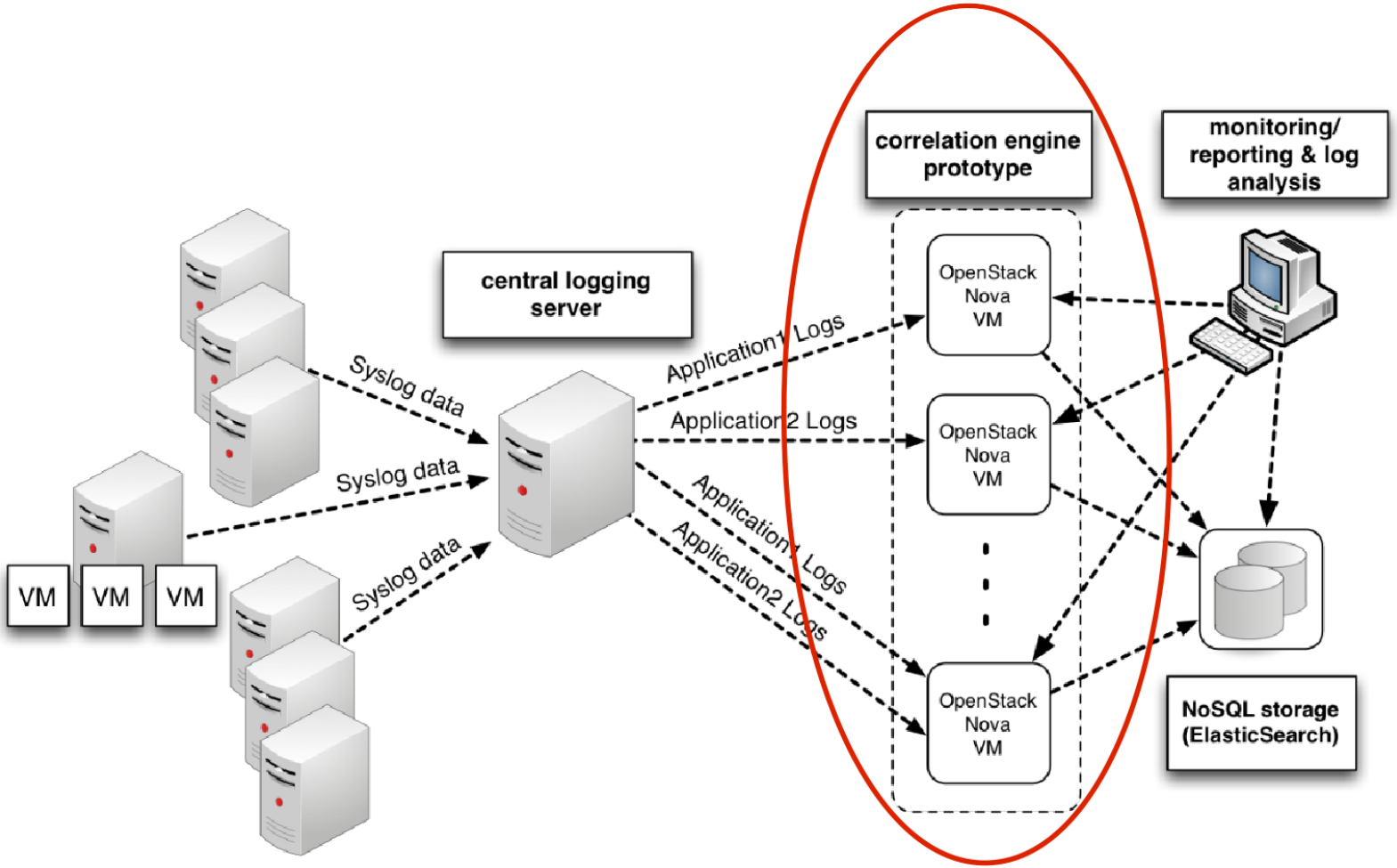
Zusammenfassung der gleichen Meldung aus unterschiedlichen Quellen

```
rule=SSHSUCCESS : Accepted password for %user:
word% from %ip:ipv4% port %port : number% %protocol:word%

rule=SSHFAILURE : Failed password for %user:
word% from %ip:ipv4% port %port : number% %protocol:word%

rule=SSHFAILURE : Failed password for invalid user %user:
word% from %ip:ipv4% port %port : number% %protocol:word%
```

liblognorm - Normalisierung

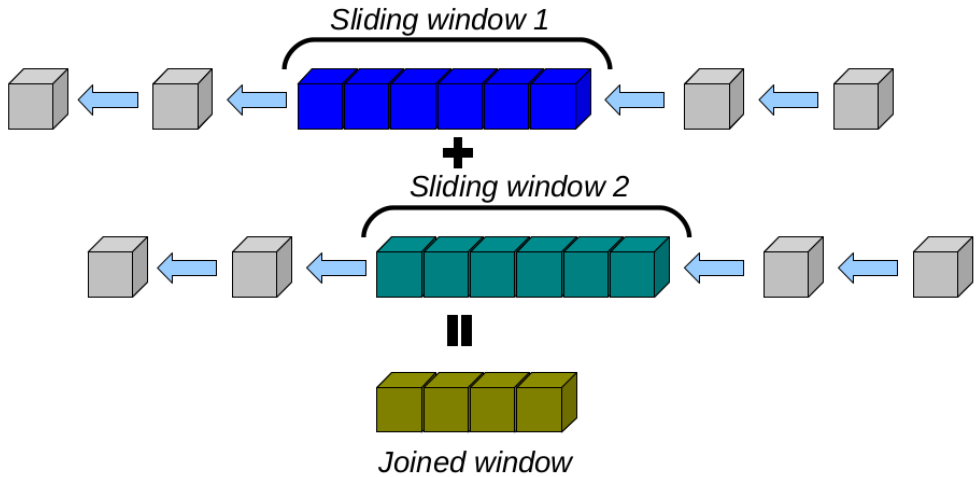
Erstellung strukturierter Daten und Weiterleitung an Korrelierungsinstanz

```
{   "data": {
        "protocol":"ssh2",
        "port" : "54548",
        "ip": "10.0.23.4",
        "user": "root"
    },
    "time":"2014-01-29T16:06:00.000",
    "host":"test.example.com",
    "facility":"auth",
    "severity":"info",
    "program" :"sshd",
    "message":" Failed password for root from
                10.0.23.4 port 54548 ssh2",
    "tags" : ["SSHFAILURE"] }
```


correlation engine prototype
monitoring/ reporting & log analysis
central logging server
Syslog data
Syslog data
Syslog data
VM
VM
VM
Application1 Logs
Application2 Logs
Application1 Logs
Application2 Logs
OpenStack Nova VM
OpenStack Nova VM
OpenStack Nova VM
NoSQL storage (ElasticSearch)

Drools-Fusion

- Complex Event Processing (CEP) Engine
- Regeln basieren auf AL
- Zeitliche Schlussfolgerungen
- Datenbestand: in-memory-engine



Quelle: Tihomir Surdilovic, RedHat (Slideshare)

```
rule "SSH brute-force attempt"
no-loop
when
    Message (   $host:host,
                $user:data["user"])
    $atts: CopyOnWriteArrayList(size >= 10)
        from collect(
            Message(    tags contains"SSHFAILURE",
                        host == $host,
                        data["user" ] == $user)
            over window : time (1m))
then
    Message last = (Message) $atts.get($atts.size( )-1) ;

    for (Object f: $atts) {
        retract ( f ) ;
}

    insert (messageFactory(last)
        . setTime(last.getTime( ))
        . setSeverity(Message.Severity.WARNING)
        . setFacility(Message.Facility.SECURITY)
        . setMessage("SSH brute-force attack" +
            "for @{data.user} from @{data.ip}")
        . addTag ("BRUTEFORCE")
        . message( )) ;
end
```

- betrachtet werden: Syslog-tags
- Trifft zu: wenn während einer Brute-Force-Attacke Login gelingt
- SSHSUCCESS innerhalb von 10s nach SSHFAILURE **und** BRUTEFORCE **und** gleicher host, user

```
rule "Successful SSH brute-force attack"
no-loop
when
     $ att: Message (    tags contains "SSHFAILURE",
                         tags contains "BRUTEFORCE",
                         $host: host ,
                         $user: data ["user"])
     $ suc: Message (    host == $host,
                         data ["user" ] == $user,
                         tags contains "SSHSUCCESS",
                         this finishes[10 s] $att)
then
     $att.addTag("INCIDENT");
     $att.setSeverity(Severity.EMERGENCY);
     $att.setMessage($att.getMessage( ) + "[bruteforce]");
update ($att);
end
```

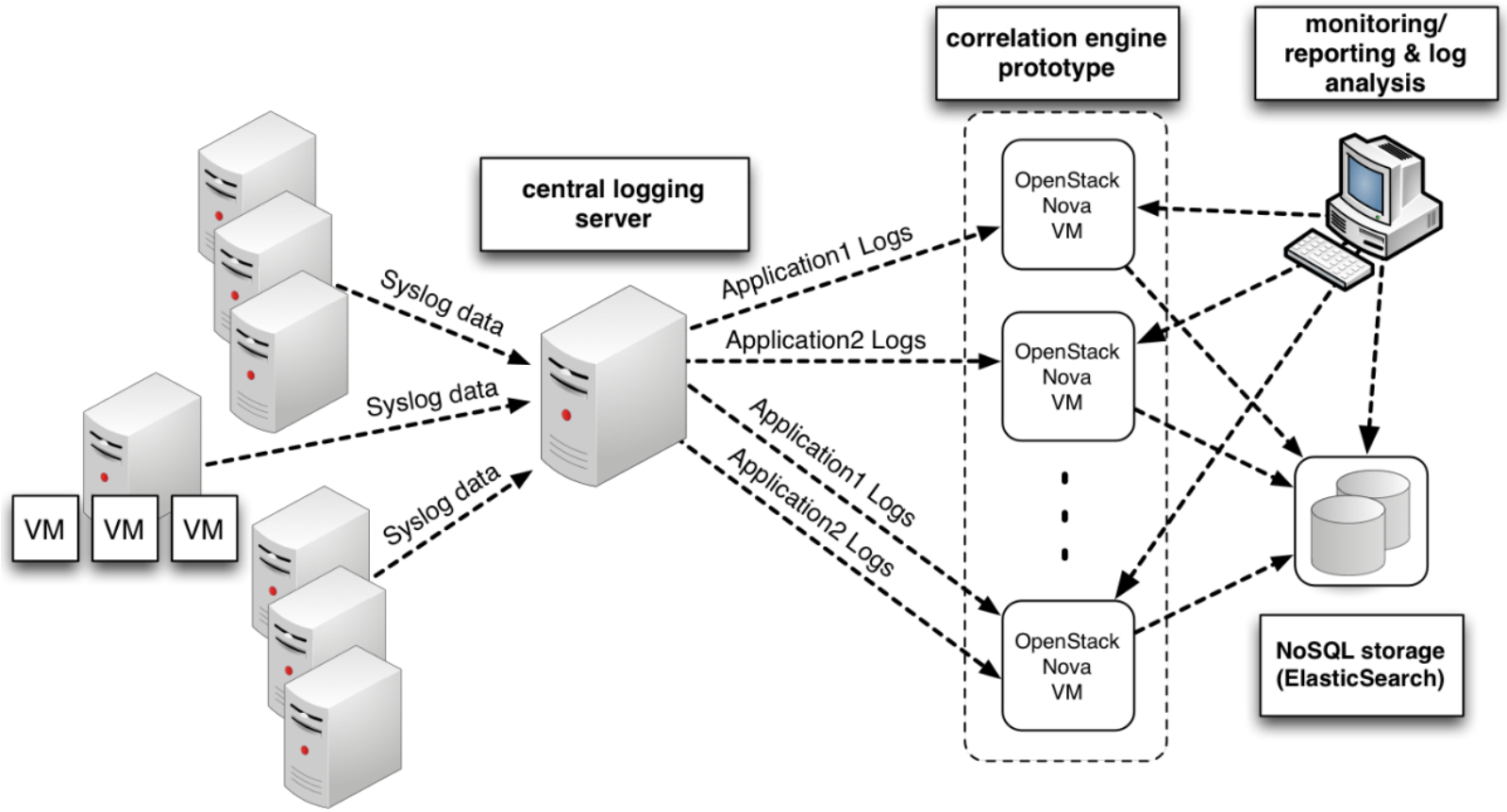
- Erzeugung neuer Syslog-Nachrichten durch Regeln
- Setzen unterschiedlicher `severity` (`Warning`: 4; `Emergency`: 0)
- Auswertung über Visualisierung, `SNMP-trap`, `check_drools`

Severity	Facility	Message
emerge	security	Accepted password for root from 10.0.23.4 port 54548 ssh2 [bruteforce]
info	auth	Failed password for root from 10.0.23.4 port 54548 ssh2
warn	security	SSH brute-force attack for root from 10.0.23.4
info	auth	Connection closed by 10.0.23.4 [preauth]
info	auth	Failed password for root from 10.0.23.4 port 54548 ssh2
info	auth	Failed password for root from 10.0.23.4 port 54548 ssh2
info	auth	Failed password for root from 10.0.23.4 port 54548 ssh2

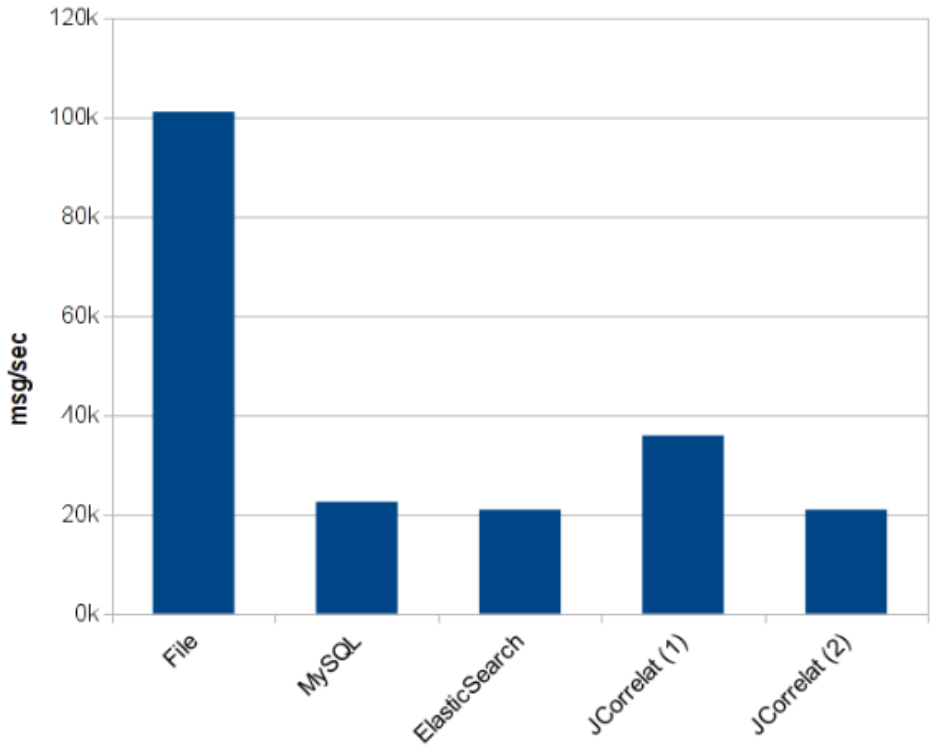
Forschungsziel: `jCorrelat` soll *correlation template engine* werden.

Korrelation

Performance- und Skalierungsbetrachtung



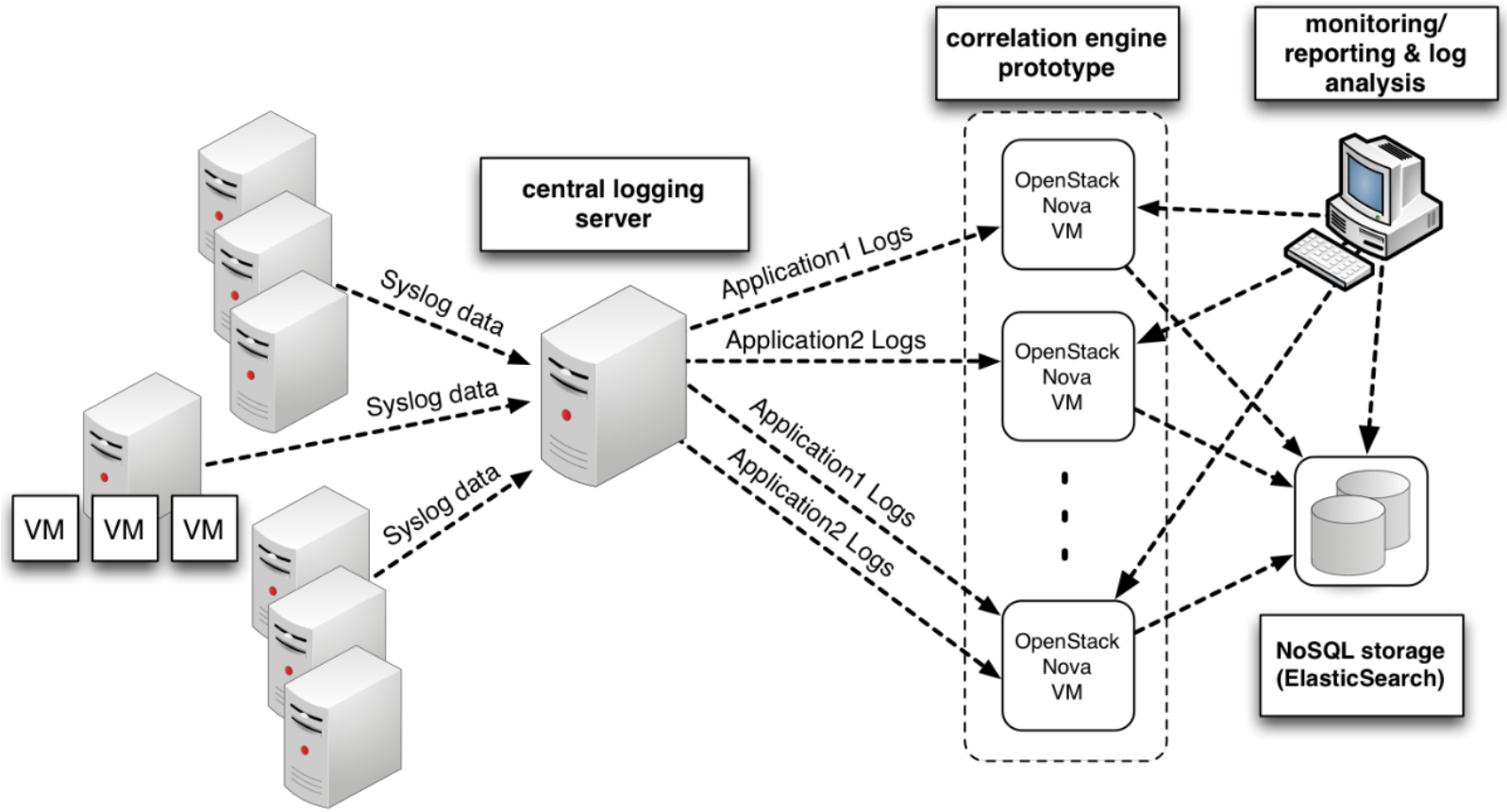
Benchmark



Limitierender Faktor: *in-memory-engine*

Korrelation

Performance- und Skalierungsbetrachtung



scanning detection systems

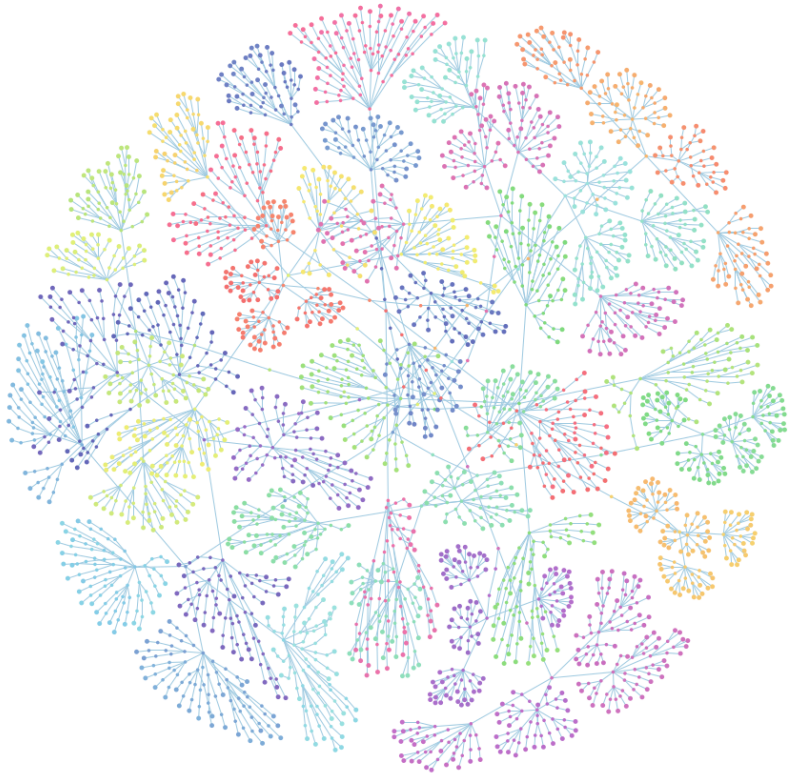
- Trafficanalyse zu ungenau
- Erstellung eines *fingerprints* kaum möglich

Verkehrskorrelation durch statistische Verfahren

- Scans werden zeitlich korreliert
- Scan-Traffic Zuordnung zu Scan-Technik
- Identifizierung von *orchestrated probing*

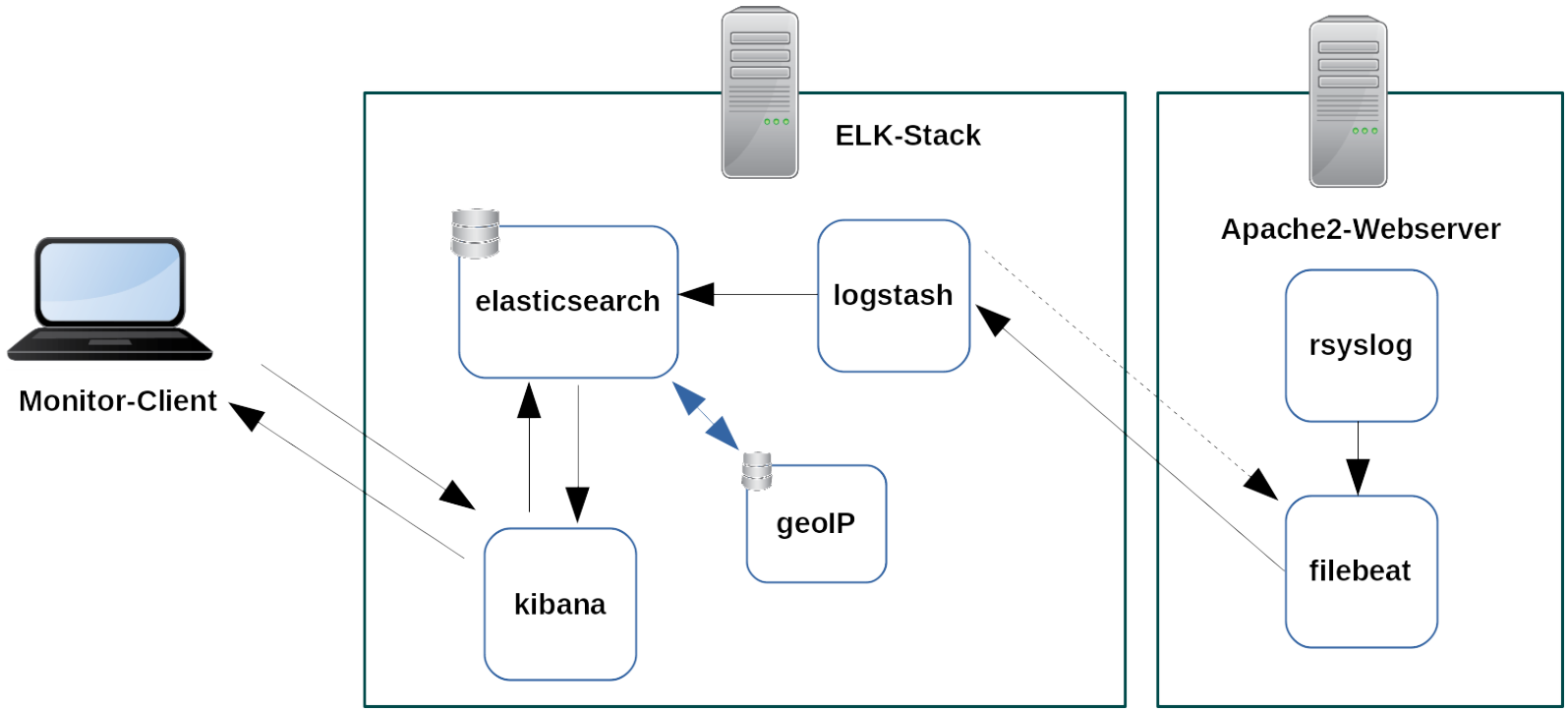
Probing

Korrelierung von Netzwerkscans



Quelle: Bou-Harb, E., Debbabi, M., & Assi, C. (2014) Behavioral analytics for inferring large-scale orchestrated probing events. In 2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS) (pp. 506–511). New York, NY: IEEE.

ELK: **E**lasticsearch, **L**ogstash, **K**ibana



ELK-Server

Danke für die Aufmerksamkeit

Folien stehen auf *github.com* zur Verfügung:

`github.com/meetunix/seminar-komplex`

`martin.steinbach@uni-rostock.de`