AUSTRIAN INSTITUTE OF TECHNOLOGY
TOMORROW TODAY

ÆCID: A SELF-LEARNING ANOMALY DETECTION APPROACH BASED ON LIGHT-WEIGHT LOG ANALYTICS

BSides Vienna, 30/11/2019

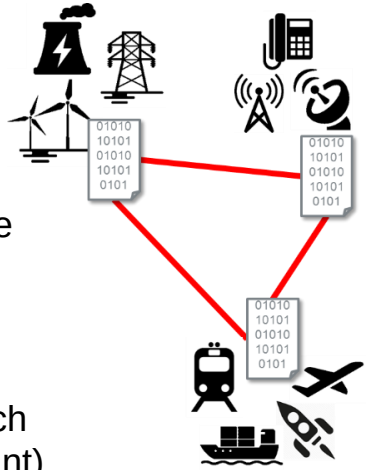
Markus Wurzenberger
Max Landauer

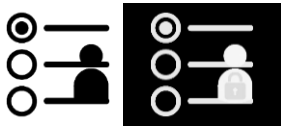
OVERVIEW

- **Theory:**
  - Motivation & challenges
  - AECID/AMiner approach
    - Concept
    - Log parsing, anomaly detection, parser generation
    - Design and architecture
- **Demonstration**
  - AECID-PG: Applied to journal messages (journald)
  - AMiner: Training phase
  - AMiner: Detection phase
    - Whitelisting
    - Selected detectors
- **Outlook**

MOTIVATION FOR ANOMALY DETECTION

- IoT, CPS and Industry 4.0 lead to an increasing **interconnection** between the **physical and digital world**
  - **No central understanding of complete systems**: experts only for layers, components and specific technologies
  - **Different timeline of life cycles (IT/OT):** standard enterprise server software with frequent updates, safety-critical systems get less frequent updates
  - Mix of technologies monitored and controlled only with a **mix of security solutions**
  - Each component offers **numerous configuration or operation modes,** much more than humans can understand or comprehend (but only a few are relevant)
  - Therefore **numerous unknown attack modes (vectors)** exist, humans cannot describe them all for **signature based blacklisting approaches**
- Novel white-list approaches that model a baseline behavior and **discover deviations from normal system behavior** are required.





ATTACK(ER) CHARACTERISTICS 2019

- Attacks do **not** rely on **technical exploits only**
  - User opens mail attachment, executes malware
  - Theft of passwords, web service cookies
  - Gain access to accounts with high privilege level
  - Exploit configuration errors

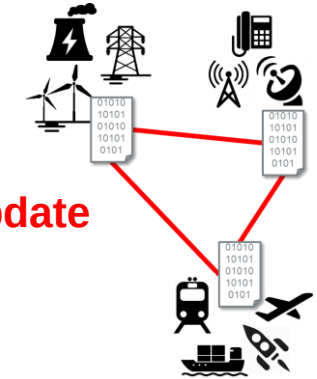
- **Atypical use** of systems
  - Access other DMZ servers from a compromised web server
  - Use of telnet-maintenance interface instead of the web interface
  - Login using backup system SSH key intended for SFTP file transfer
- **Anomalies cannot be flagged** as clearly malicious **looking at single systems only**
  - Single system activities look sane: admin authenticates to webserver, performs configuration changes (POST /Change.php)…
  - … but usually this will occur only from the administrator‘s machine, not from a secretary’s computer nor from the printer

MOTIVATION & CHALLENGES

- Pure **blacklist** detection techniques based on signatures are **NOT sufficient**

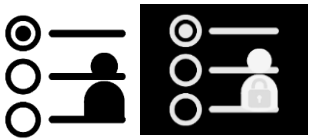
Only known attacks detectable

Fast changing cyber threat landscape → Difficult to **timely update** signatures and rules



- Paradigm shift towards **anomaly-based detection**
  - **Whitelist** approaches for **normal** system **behavior modelling**
  - Detection of **unknown attacks**

- **NO general applicable** intrusion detection solution (FN/FP)
- End-to-end encryption → **insufficient** to monitor network traffic only
- **AIT'S SOLUTION:** Analyze textual **log data** (e.g., syslog)



LOG LINES

ntpd [16721]: Listen and drop on 0 v4wildcard 0.0.0.0 UDP 123
ntpd [16721]: Listen and drop on 1 v6wildcard :: UDP 123
ntpd [16721]: Listen normally on 2 lo 127.0.0.1 UDP 123
ntpd [16721]: Listen normally on 3 eth0 134.74.77.21 UDP 123
ntpd [16721]: Listen normally on 4 eth1 10.10.0.57 UDP 123
ntpd [16721]: Listen normally on 5 eth1 fe80::5652:ff:fe5a:f89f UDP 123
ntpd [16721]: Listen normally on 6 eth0 fe80::5652:ff:fe01:1fff UDP 123
ntpd [16721]: Listening on routing socket on fd #24 for interface updates

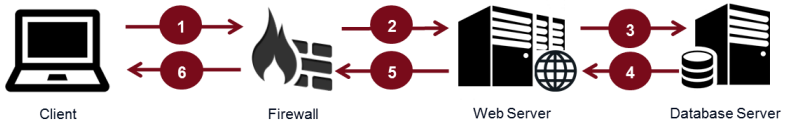
OUR APPROACH – ÆCID!

Most solutions work on the **network** (netflows, DPI), however we **inspect the host**

- End-to-End encryption (tunneling) avoids DPI
- Virtualization – machines on the same hypervisor are hard to monitor
- Verbose Log data contains more expressive events than single packets

Automatic Event Correlation for Incident Detection

- Keeps track of system **events**, their **dependencies**, their **occurrences**
  - Analyzes sequentially produced textual **log data** (e.g., syslog) that reflect actual system events
  - Dynamically learns the **normal system (utilization) behavior model**
  - **Detects** deviations from that model



AECID DETECTABLE ANOMALIES

- **Point anomalies**
  - Client access with unknown user agent (e.g., Internet Explorer instead of Firefox)
  - **Whitelisting:** only Firefox is allowed, any other triggers an alarm
  - **Blacklisting:** list of prohibited tools, vulnerable to incompleteness
- **Anomalous event parameter (combinations)**
  - E.g., access outside working hours
- **Anomalous event frequency**
  - E.g., data theft: unusual number of database accesses from a single client in a short time-window
- **Anomalous event sequence**
  - E.g., SQL-Injection: Access-chain violation:
    - Firewall/Webserver/Database-Server

AECID CONCEPT

- **Online log based anomaly detection:**
  - Monitor any (unstructured) **textual event data** (e.g., syslog, windows event log)
  - **Self-learning** and **whitelisting** → no attack signatures required
  - No semantic interpretation → only constant syntax
  - Automatic detection of **relevant log parts**

- → **Flexible** and **domain-independent** general applicable solution:
  - **Network**-, **application**- and **cross-layer** usage
  - Coverage of legacy systems, systems with small market shares and poor documentation → **no signatures and parsers** exist

- Prerequisite: logging

THE ÆCID APPROACH

1. **Log parser** generation
   - A “recipe” on how to dissect log lines of unknown grammar
   - Make log data usable for analysis → structured representation & easy to access
2. **Hypotheses** proposal
   - Distribution of property values (e.g., IP addresses, user names, …) in single events
   - And across multiple events
   - Correlation of event types
3. **Rule** generation through continuous hypotheses evaluation
   - Sort out unstable hypotheses and create rules for stable ones
   - Constitution of the system behavior model (learned behavior model)
4. **Anomaly detection**: rate the deviation of actual system behavior from the learned behavior model (anomalous points / context / frequency / sequence of events)

All steps take place in parallel, i.e., even during the anomaly detection phase, new hypotheses are created on the fly.

STEP 1: PARSING – FAST LOG DATA PROCESSING

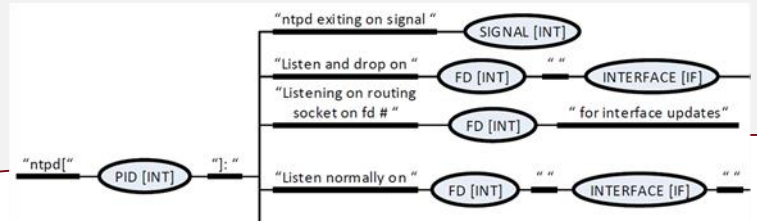
- **Parser model:** describes system behavior
  - **Loglines** represented as **tree-like graph (O(log(n))** → Parse data once!
  - **No regex (O(n))** for whole line required → **fast line processing**, rule evaluation
  - → **Online Anomaly Detection**
  - Describe information most efficiently – with **minimal storage requirements**
  - **Efficient** log line **classification**

```
Dec 15 00:10:27 www0.some.domain apache: 30086 10.0.0.1:80 "www.seite.at"
"www.seite.at" 192.168.0.1 - - [15/Dec/2015:00:10:27 +0000] 126 "GET / HTTP/1.1" 302
212 "-" „Monitoring Agent„
```

/model/syslog/time: 2015-12-15 00:10:27

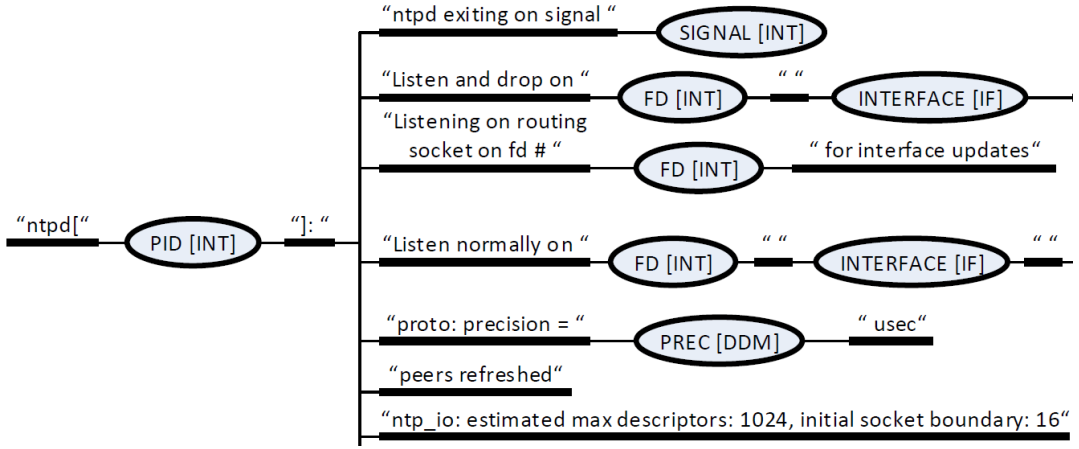
/model/syslog/host: www0.some.domain

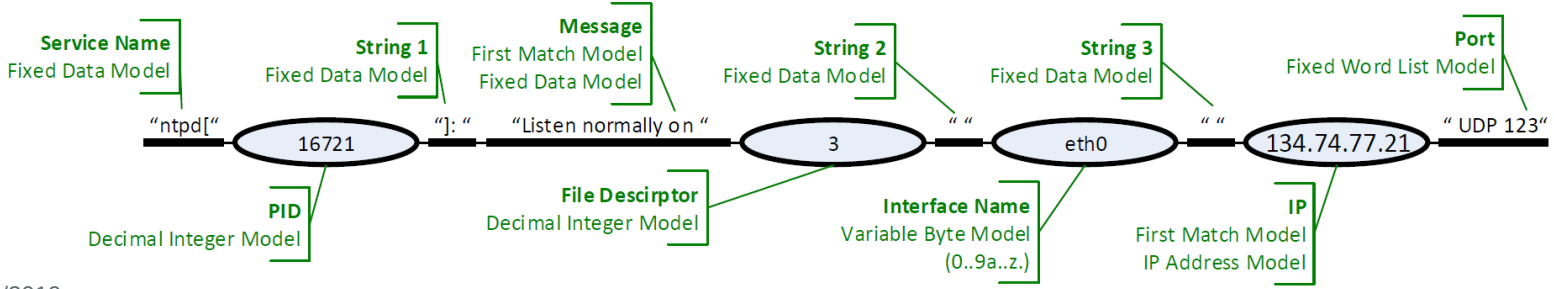
/model/services/apache/sname: apache



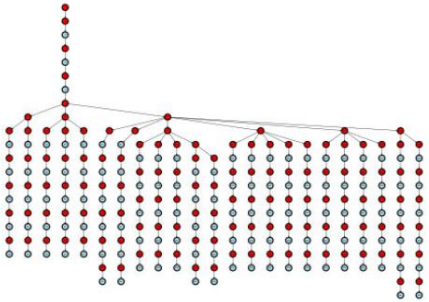
APPROACH – STEP 1: PARSING

```
ntpd[16721]: Listen and drop on 0
   v4wildcard 0.0.0.0 UDP 123
ntpd[16721]: Listen and drop on 1
   v6wildcard :: UDP 123
ntpd[16721]: Listen normally on 2 lo
   127.0.0.1 UDP 123
ntpd[16721]: Listen normally on 3
   eth0 134.74.77.21 UDP 123
ntpd[16721]: Listen normally on 4
   eth1 10.10.0.57 UDP 123
```

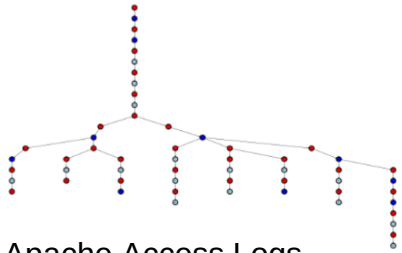




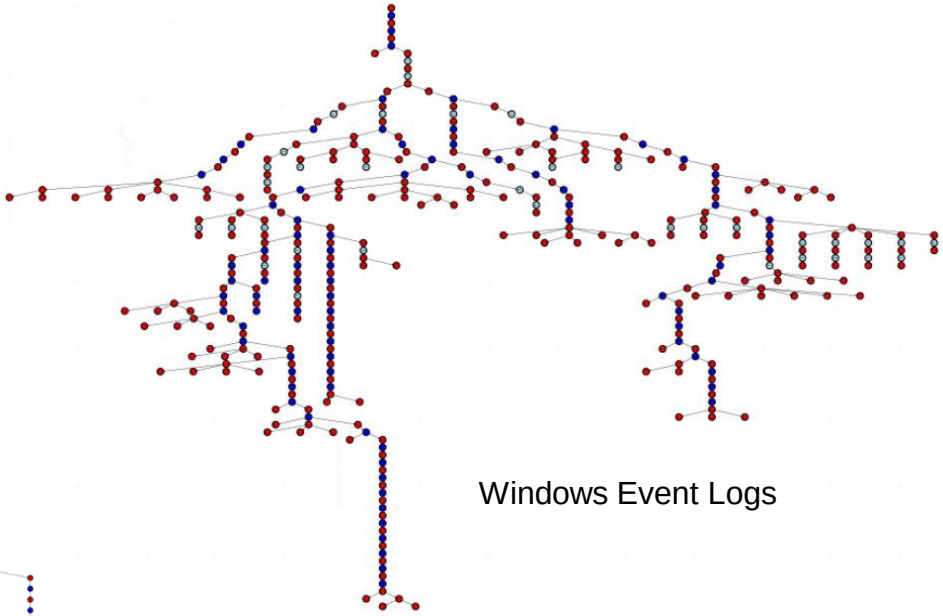
PARSER TREES: REAL-WORLD EXAMPLES



Suricata Logs

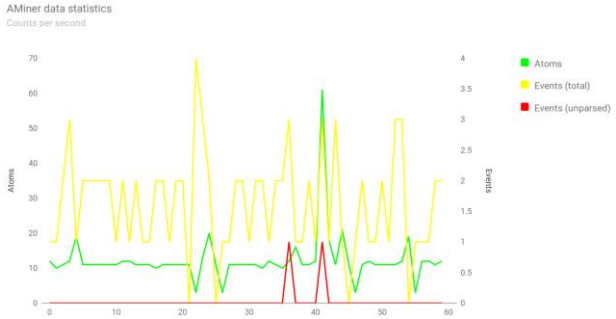


Apache Access Logs

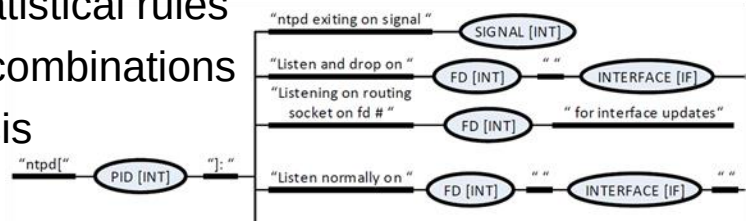


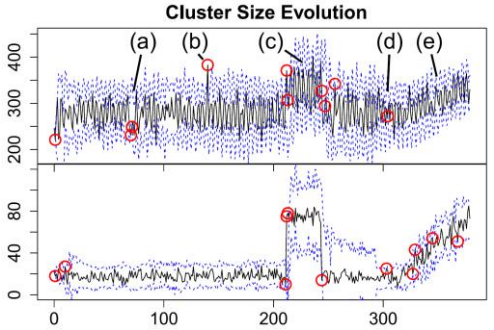
Windows Event Logs

AECID DETECTION MECHANISMS



- **Whitelisting** to overcome limitations of blacklisting
  - Detect unknown patterns
  - Signatures can be evaded by modification of attacks
  - Blacklists only flag clearly malicious behavior:
    - **Behavior acceptable in one context**, e.g. system update replaces files, **is clearly abnormal in different context**, e.g. graphic card firmware loading component replaces files
    - Normal system operation, **only some system states are encountered** even if plenty of system states (rare states, error states) are possible
- **Rule-based** detection to monitor complex system processes
  - Correlation and statistical rules
  - Values and value combinations
  - Time series analysis





APPROACH – STEP 2: HYPOTHESES PROPOSAL (1/2)

```
Jun 20 00:59:37 localhost sshd[1008]: Accepted public key for backup
from 172.29.147.33 port 54149 ssh2: RSA SHA256:9k...
```

/model/syslog/time: **Jun 20 00:59:37**
/model/syslog/host: **localhost**
/model/services/sshd/sname: **sshd**
/model/services/sshd/msg/acceptedpk/pid: **1008**
/model/services/sshd/msg/acceptedpk/user: **backup**
/model/services/sshd/msg/acceptedpk/originip: **172.29.147.33**
/model/services/sshd/msg/acceptedpk/port: **54149**
/model/services/sshd/msg/acceptedpk/protocol: **ssh2**
/model/services/sshd/msg/acceptedpk/crypto: **RSA**
/model/services/sshd/msg/acceptedpk/fingerprint: **SHA256:9k...**

Simple Example Hypotheses:

```
user{backup} ~ remoteip{172.29.147.33}
user{backup} ~ fingerprint{SHA256:9k...}
user{backup} only allowed in time_hh{[00,03]}
...
```

- Different Methods for hypothesis generation (incl. brute force)
- Coverage of events is complex to determine
- Maximize detection capabilities with minimum number of (stable) hypotheses
- Continuous learning in parallel to detection

APPROACH – STEP 2: HYPOTHESES PROPOSAL (2/2)

Firewall Logs

permitted HTTP traffic sourced from inside (eth1) with NAT (Check Point FW/VPN 1)

```
Dec 15 09:10:26 accept
www0.some.domain >eth1 product VPN-1 &
Firewall-1  src 10.0.0.1 s_port 45213
dst 192.168.0.1 service http proto
tcp  xlatesrc 192.168.0.10 rule 5
```

Web Server logs

Ressource retrieval via HTTP on Apache Webserver

```
Dec 15 09:10:27 www0.some.domain
apache: 30086 192.168.0.1:80
"www.page.at" "www.page.at"
192.168.0.10 - - [15/Dec/2015:09:10:27
+0000] 126 "GET / HTTP/1.1" 302 212 "-
" "Mozilla/5.0"
```

Cross-System Example Hypotheses:

- event „HTTP retrieval“ on Apache with parameters „**www.page.at**“ conditions „permit HTTP“ from src={**10.0.0.1, ...**} on FW in a time window of 5000ms
- src={**192.168.0.10**} in „HTTP retrieval“ ~ src={**10.0.0.1**} in „permit HTTP“ in a time window of 5000ms
- ...

AECID - ARCHITECTURE

AMiner

- Lightweight base implementation
- Parses log lines
- Verifies rules
- Triggers alarms
- License: open source

launchpad

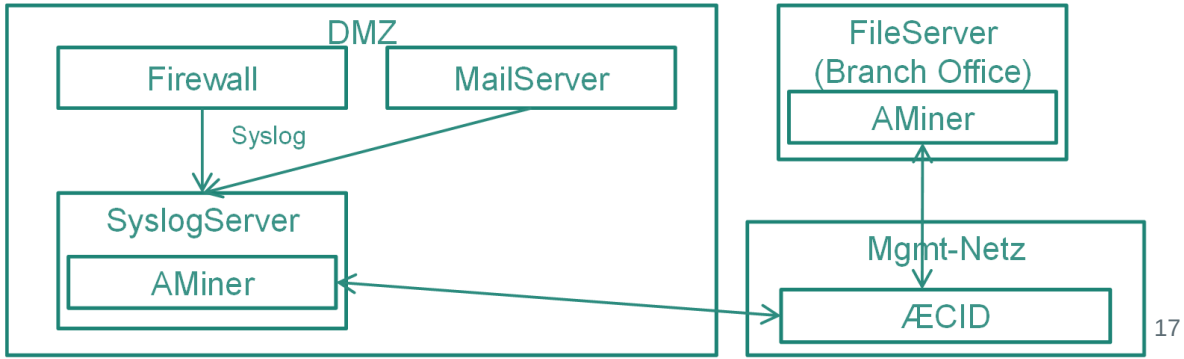
https://git.launchpad.net/logdata-anomaly-miner

debian

AECID Central

- Intelligent control center
- Receives unknown log lines from AMiner instances
- Distributes and adapts system model and rule-set
- Research prototypes

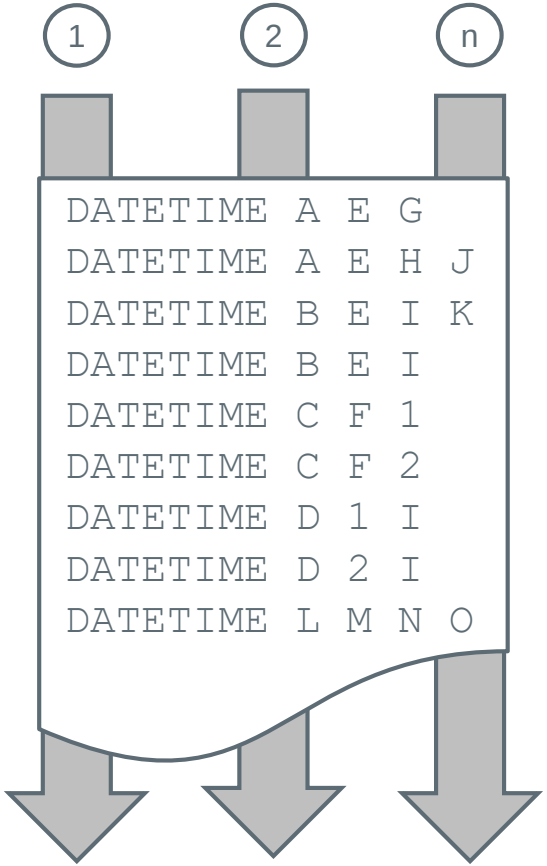
https://aecid.ait.ac.at/

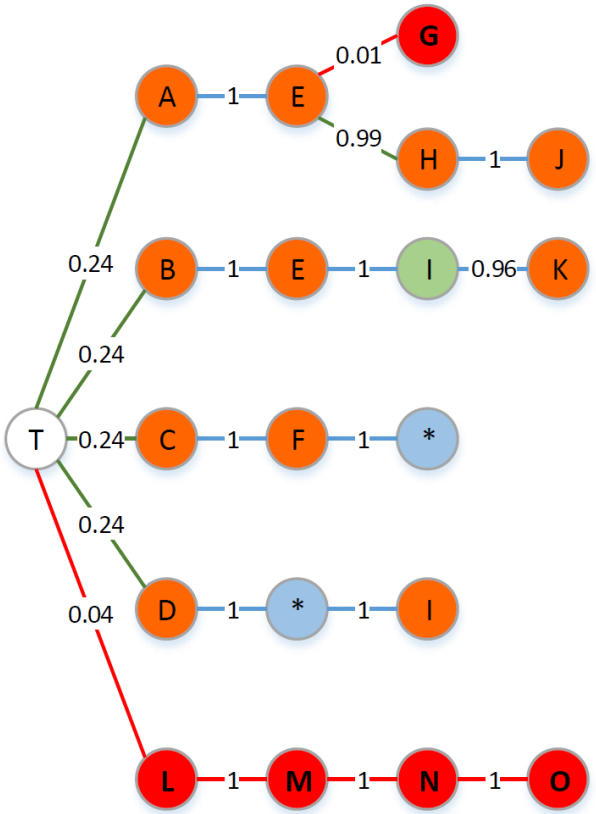


AECID-PG: PARSER GENERATOR

- $\theta_1 = 0.1$
- $\theta_2 = 0.95$
- $\theta_3 = 0.9$
- $\theta_4 = 0.01$

- Legend:
  - Fixed
  - Optional
  - Variable
  - Deleted



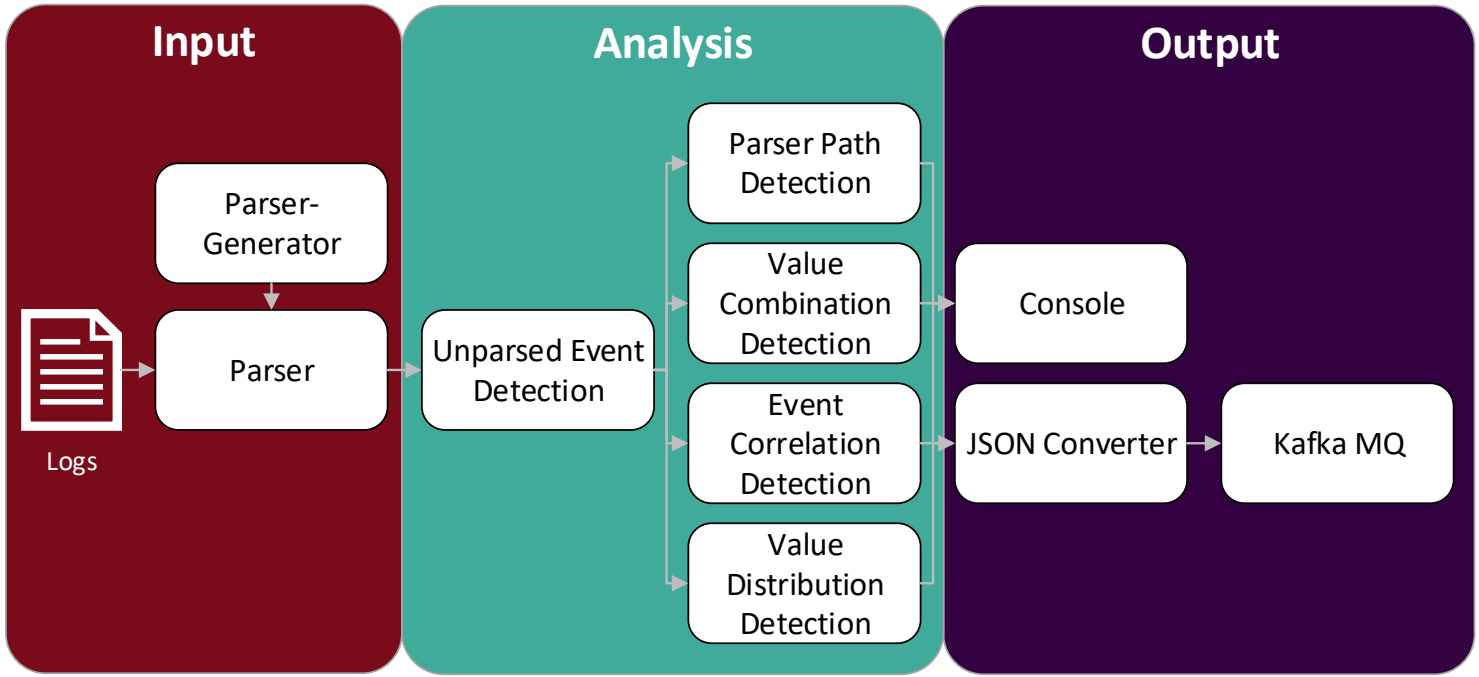


AUSTRIAN INSTITUTE OF TECHNOLOGY
TOMORROW TODAY

DEMONSTRATION

AECID + AMiner

DEMONSTRATION



SAMPLE LOGS

ntpd [16721]: Listen and drop on 0 v4wildcard 0.0.0.0 UDP 123
ntpd [16721]: Listen and drop on 1 v6wildcard :: UDP 123
ntpd [16721]: Listen normally on 2 lo 127.0.0.1 UDP 123
ntpd [16721]: Listen normally on 3 eth0 134.74.77.21 UDP 123
ntpd [16721]: Listen normally on 4 eth1 10.10.0.57 UDP 123
ntpd [16721]: Listen normally on 5 eth1 fe80::5652:ff:fe5a:f89f UDP 123
ntpd [16721]: Listen normally on 6 eth0 fe80::5652:ff:fe01:1fff UDP 123
ntpd [16721]: Listening on routing socket on fd #24 for interface updates

LOG TEMPLATES

ntpd [16721]: Listen and drop on 0 v4wildcard 0.0.0.0 UDP 123
ntpd [16721]: Listen and drop on 1 v6wildcard :: UDP 123
ntpd [16721]: Listen normally on 2 lo 127.0.0.1 UDP 123
ntpd [16721]: Listen normally on 3 eth0 134.74.77.21 UDP 123
ntpd [16721]: Listen normally on 4 eth1 10.10.0.57 UDP 123
ntpd [16721]: Listen normally on 5 eth1 fe80::5652:ff:fe5a:f89f UDP 123
ntpd [16721]: Listen normally on 6 eth0 fe80::5652:ff:fe01:1fff UDP 123
ntpd [16721]: Listening on routing socket on fd #24 for interface updates

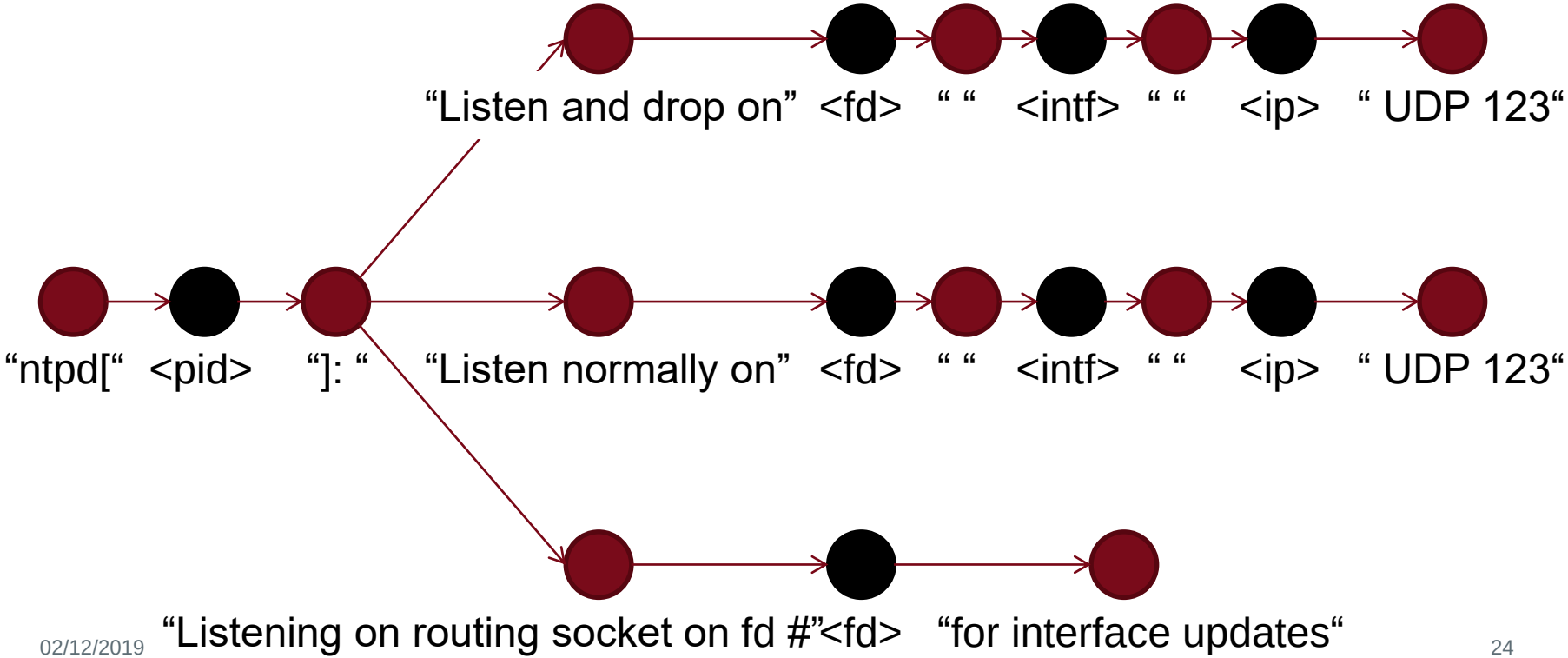
ntpd[§]: Listen and drop on § § § UDP 123
ntpd[§]: Listen normally on § § § UDP 123
ntpd[§]: Listening on routing socket on fd § for interface updates

LOG TEMPLATES

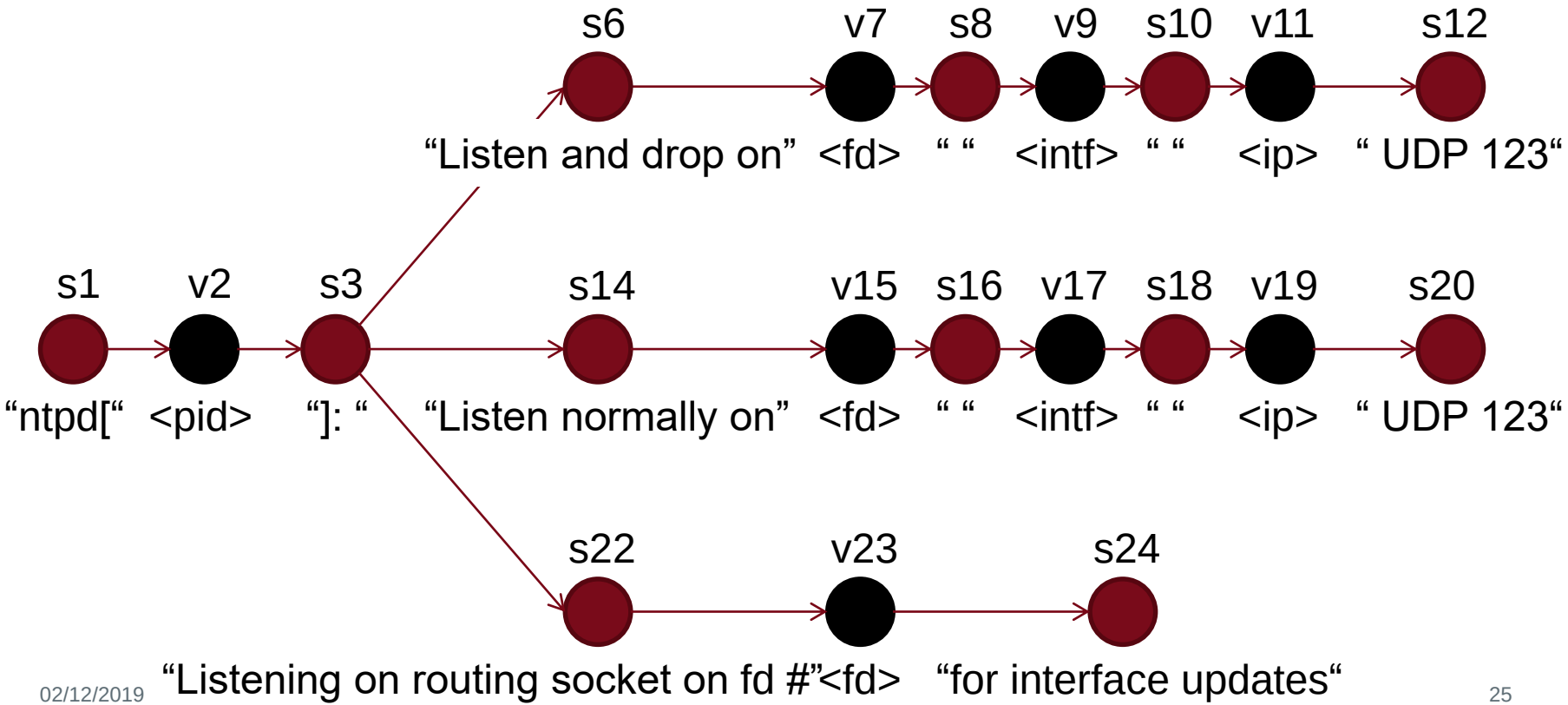
ntpd [16721]: Listen and drop on 0 v4wildcard 0.0.0.0 UDP 123
ntpd [16721]: Listen and drop on 1 v6wildcard :: UDP 123
ntpd [16721]: Listen normally on 2 lo 127.0.0.1 UDP 123
ntpd [16721]: Listen normally on 3 eth0 134.74.77.21 UDP 123
ntpd [16721]: Listen normally on 4 eth1 10.10.0.57 UDP 123
ntpd [16721]: Listen normally on 5 eth1 fe80::5652:ff:fe5a:f89f UDP 123
ntpd [16721]: Listen normally on 6 eth0 fe80::5652:ff:fe01:1fff UDP 123
ntpd [16721]: Listening on routing socket on fd #24 for interface updates

ntpd[<pid>]: Listen and drop on <fd> <intf> <ip> UDP 123
ntpd[<pid>]: Listen normally on <fd> <intf> <ip> UDP 123
ntpd[<pid>]: Listening on routing socket on fd <fd> for interface updates

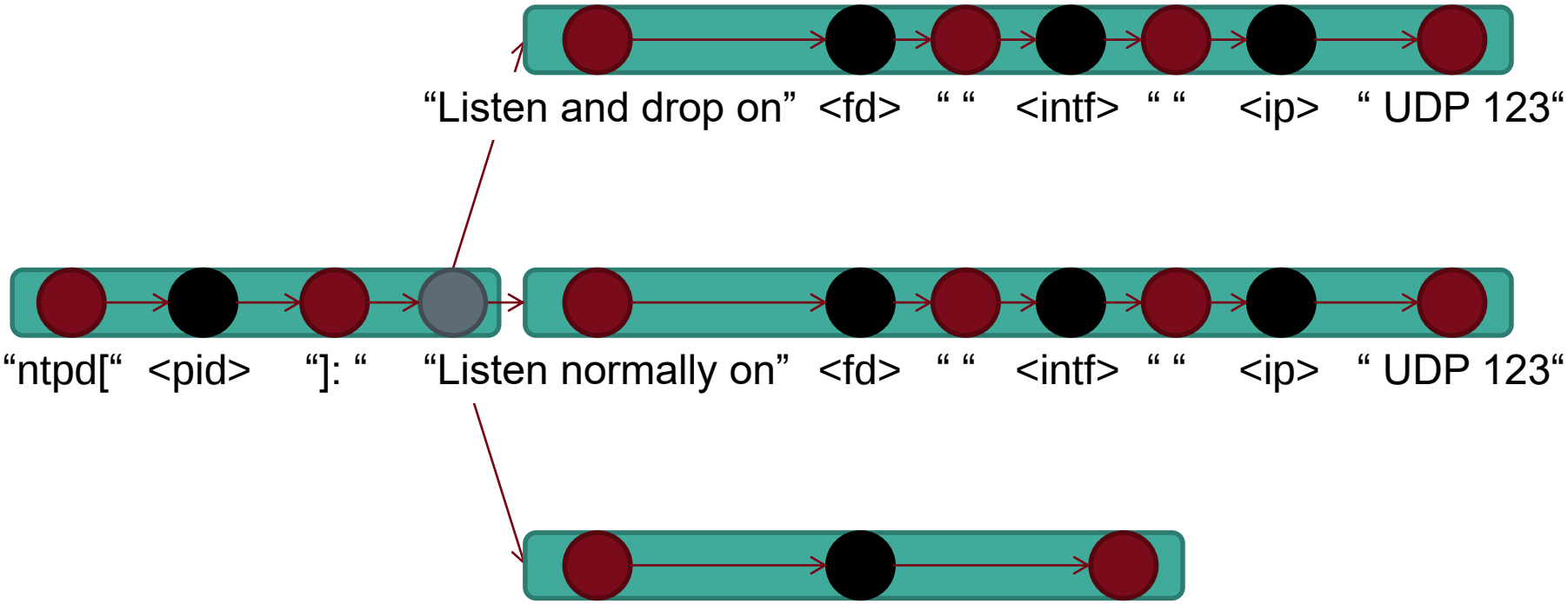
PARSER TREE


“Listen and drop on” <fd> “ “ <intf> “ “ <ip> “ UDP 123“
“ntpd[“ <pid> “]: “
“Listen normally on” <fd> “ “ <intf> “ “ <ip> “ UDP 123“
“Listening on routing socket on fd #”<fd> “for interface updates“

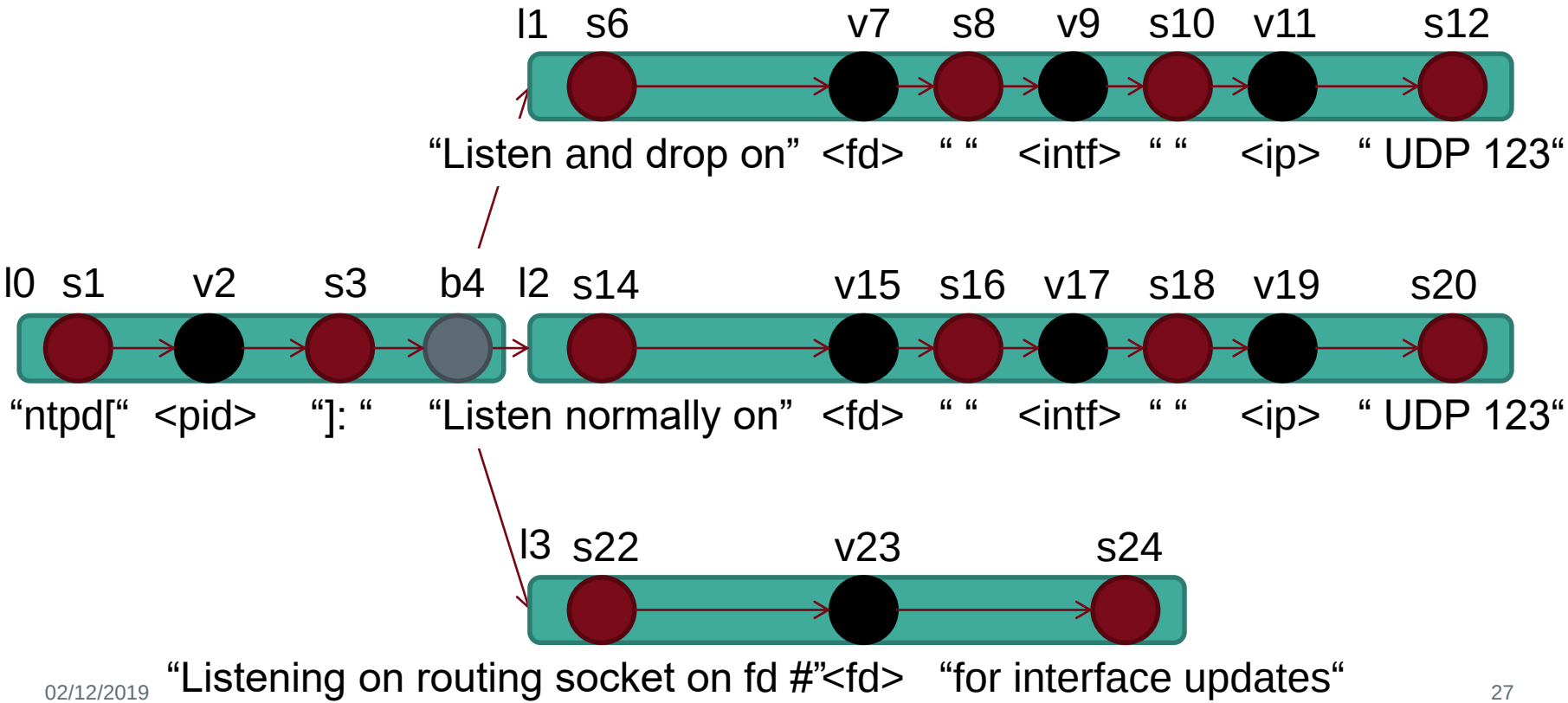
PARSER TREE


s6
v7
s8
v9
s10
v11
s12
“Listen and drop on”
<fd>
“ “
<intf>
“ “
<ip>
“ UDP 123“
s1
v2
s3
s14
v15
s16
v17
s18
v19
s20
“ntpd[“
<pid>
“]: “
“Listen normally on”
<fd>
“ “
<intf>
“ “
<ip>
“ UDP 123“
s22
v23
s24
“Listening on routing socket on fd #”<fd>
“for interface updates“

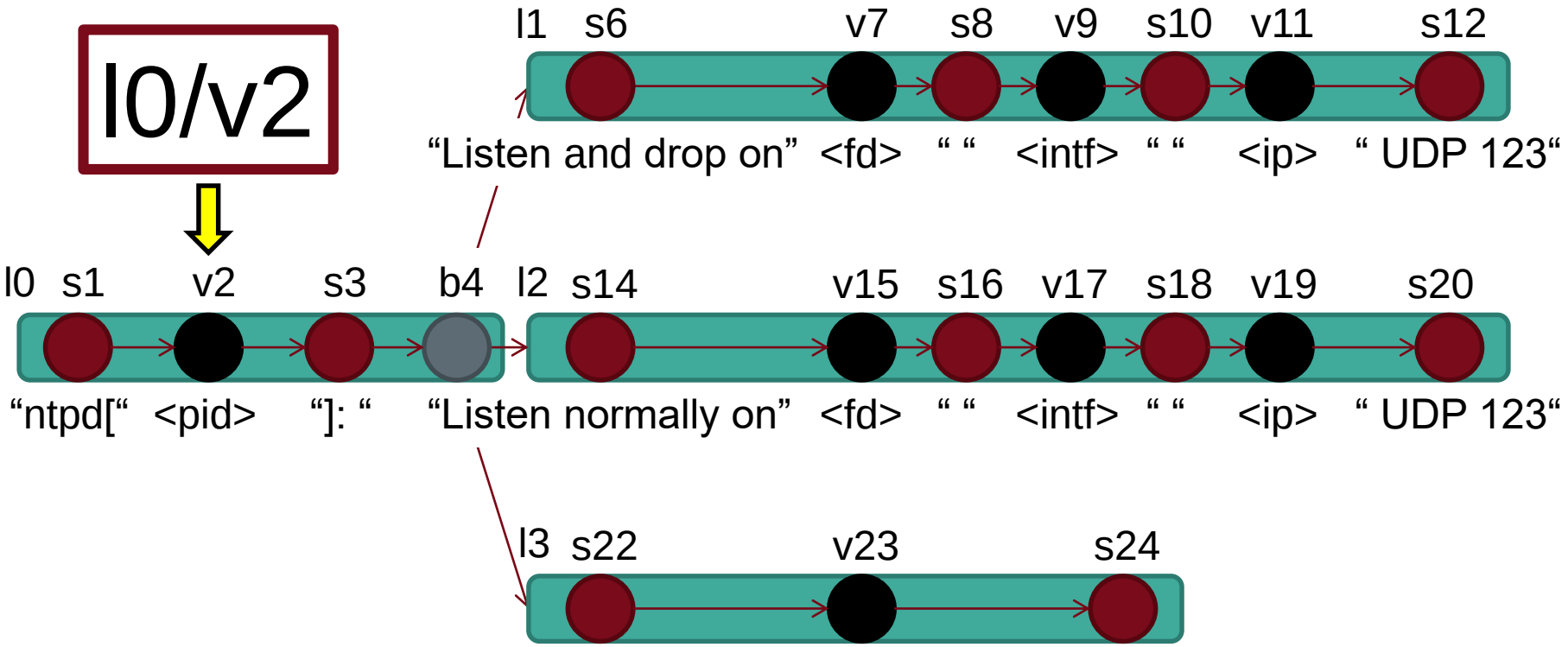
PARSER TREE


“Listen and drop on” <fd> “ “ <intf> “ “ <ip> “ UDP 123“
“ntpd[“ <pid> “]: “ “Listen normally on” <fd> “ “ <intf> “ “ <ip> “ UDP 123“
“Listening on routing socket on fd #”<fd> “for interface updates“

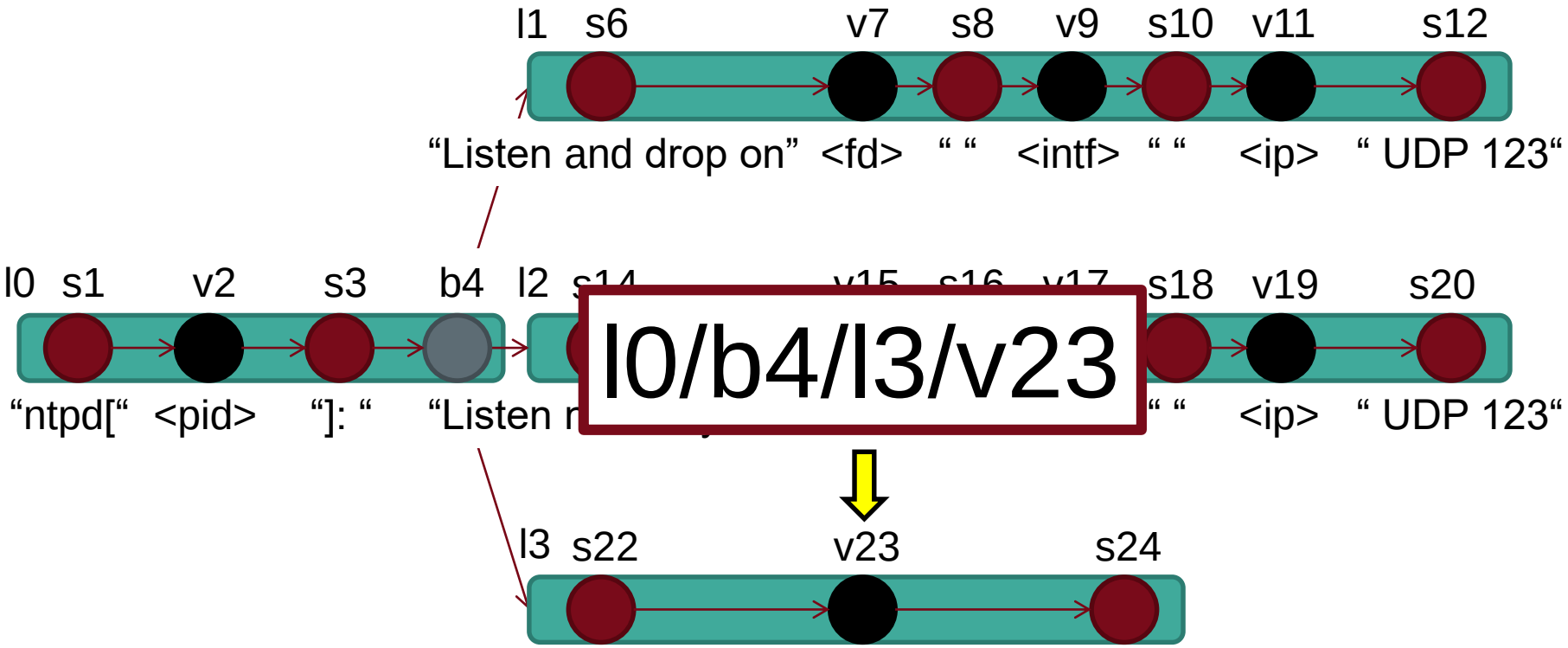
PARSER TREE


l1 s6 v7 s8 v9 s10 v11 s12
“Listen and drop on” <fd> “ “ <intf> “ “ <ip> “ UDP 123“
l0 s1 v2 s3 b4
l2 s14 v15 s16 v17 s18 v19 s20
“ntpd[“ <pid> “]: “
“Listen normally on” <fd> “ “ <intf> “ “ <ip> “ UDP 123“
l3 s22 v23 s24
“Listening on routing socket on fd #”<fd> “for interface updates“

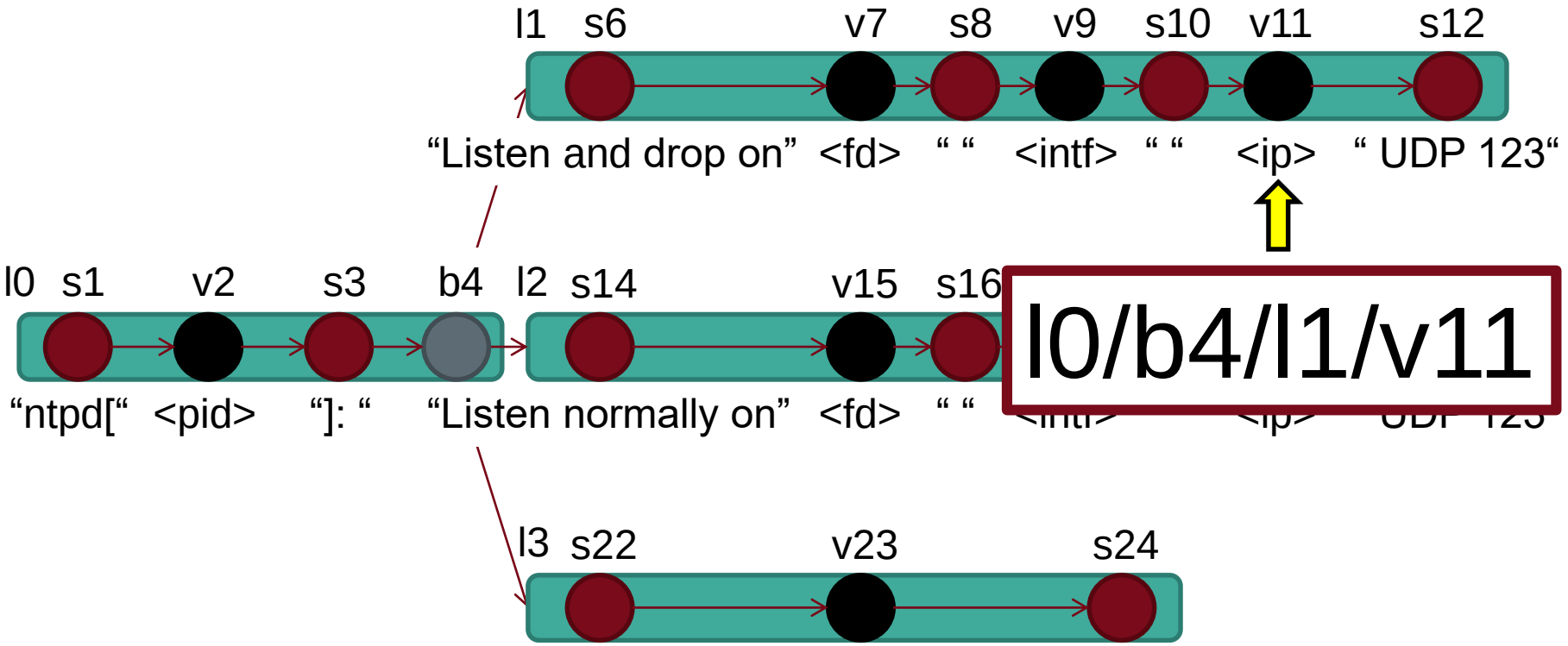
PARSER TREE


l0/v2
l1 s6 v7 s8 v9 s10 v11 s12
“Listen and drop on” <fd> “ “ <intf> “ “ <ip> “ UDP 123“
l0 s1 v2 s3 b4 l2 s14 v15 s16 v17 s18 v19 s20
“ntpd[“ <pid> “]: “ “Listen normally on” <fd> “ “ <intf> “ “ <ip> “ UDP 123“
l3 s22 v23 s24
“Listening on routing socket on fd #”<fd> “for interface updates“

PARSER TREE


l1 s6 v7 s8 v9 s10 v11 s12
“Listen and drop on” <fd> “ “ <intf> “ “ <ip> “ UDP 123“
l0 s1 v2 s3 b4 l2
“ntpd[“ <pid> “]: “ “Listen
“ “ <ip> “ UDP 123“
l0/b4/l3/v23
l3 s22 v23 s24
“Listening on routing socket on fd #”<fd> “for interface updates“

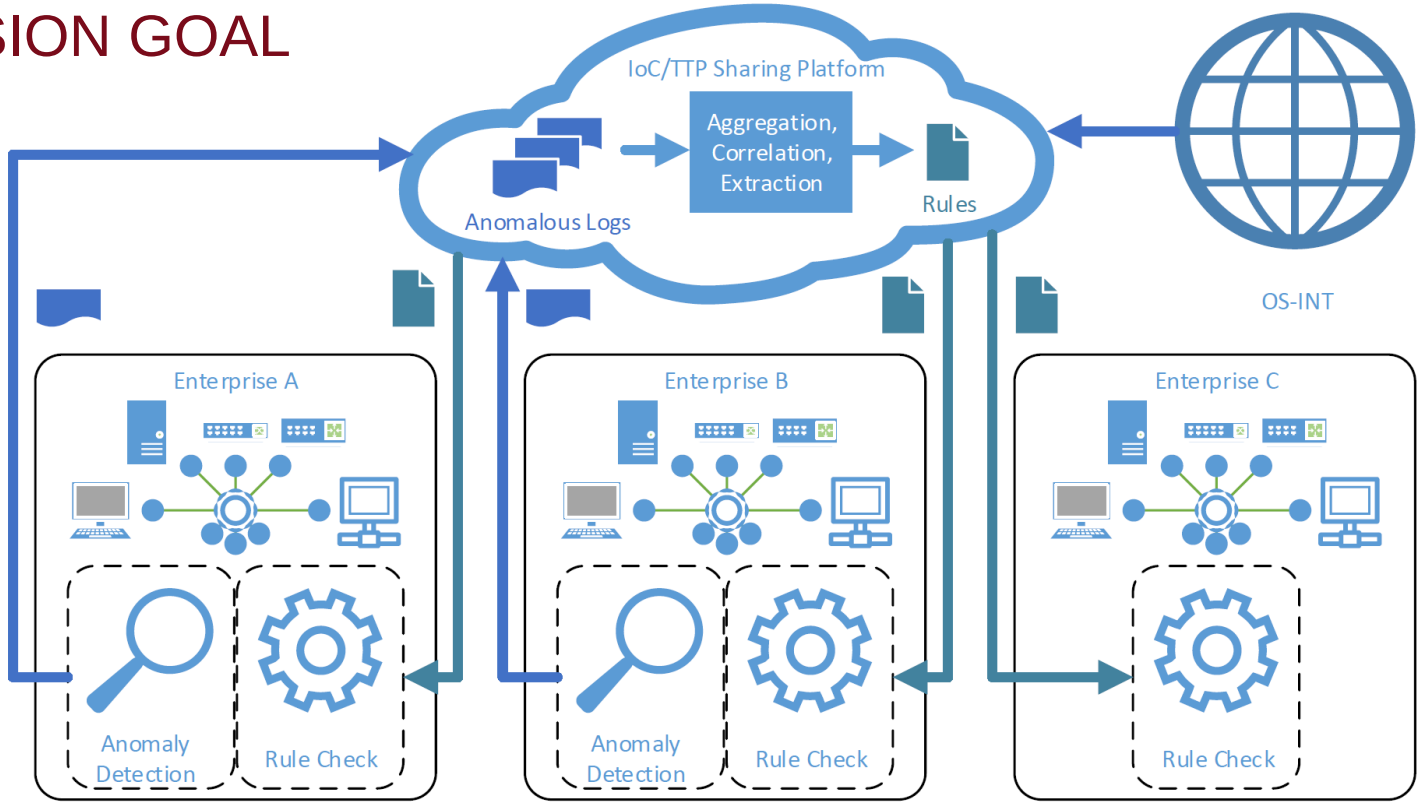
PARSER TREE


l1 s6 v7 s8 v9 s10 v11 s12
“Listen and drop on” <fd> “ “ <intf> “ “ <ip> “ UDP 123“
l0 s1 v2 s3 b4 l2 s14 v15 s16
l0/b4/l1/v11
“ntpd[“ <pid> “]: “ “Listen normally on” <fd> “ “
l3 s22 v23 s24
“Listening on routing socket on fd #”<fd> “for interface updates“

OUTLOOK

What does the future hold?

MISSION GOAL



LINKS

- **AECID:** https://aecid.ait.ac.at/
- **Demonstration Video:** https://www.youtube.com/watch?v=WhE1URkZgI8
- **AMiner** (Launchpad: Source-Code)**:** https://launchpad.net/logdata-anomaly-miner/
- **AMiner** (Debian)**:** https://packages.debian.org/sid/misc/logdata-anomaly-miner
- **Publications + Patents:** https://aecid.ait.ac.at/further-information/
- **Projects:**

AUSTRIAN INSTITUTE OF TECHNOLOGY
TOMORROW TODAY

THANK YOU!

Markus Wurzenberger, Max Landauer, 30th of November, 2019

Markus.Wurzenberger@ait.ac.at

Max.Landauer@ait.ac.at