Network Diagnostics Using Passive Network Monitoring and Packet Analysis

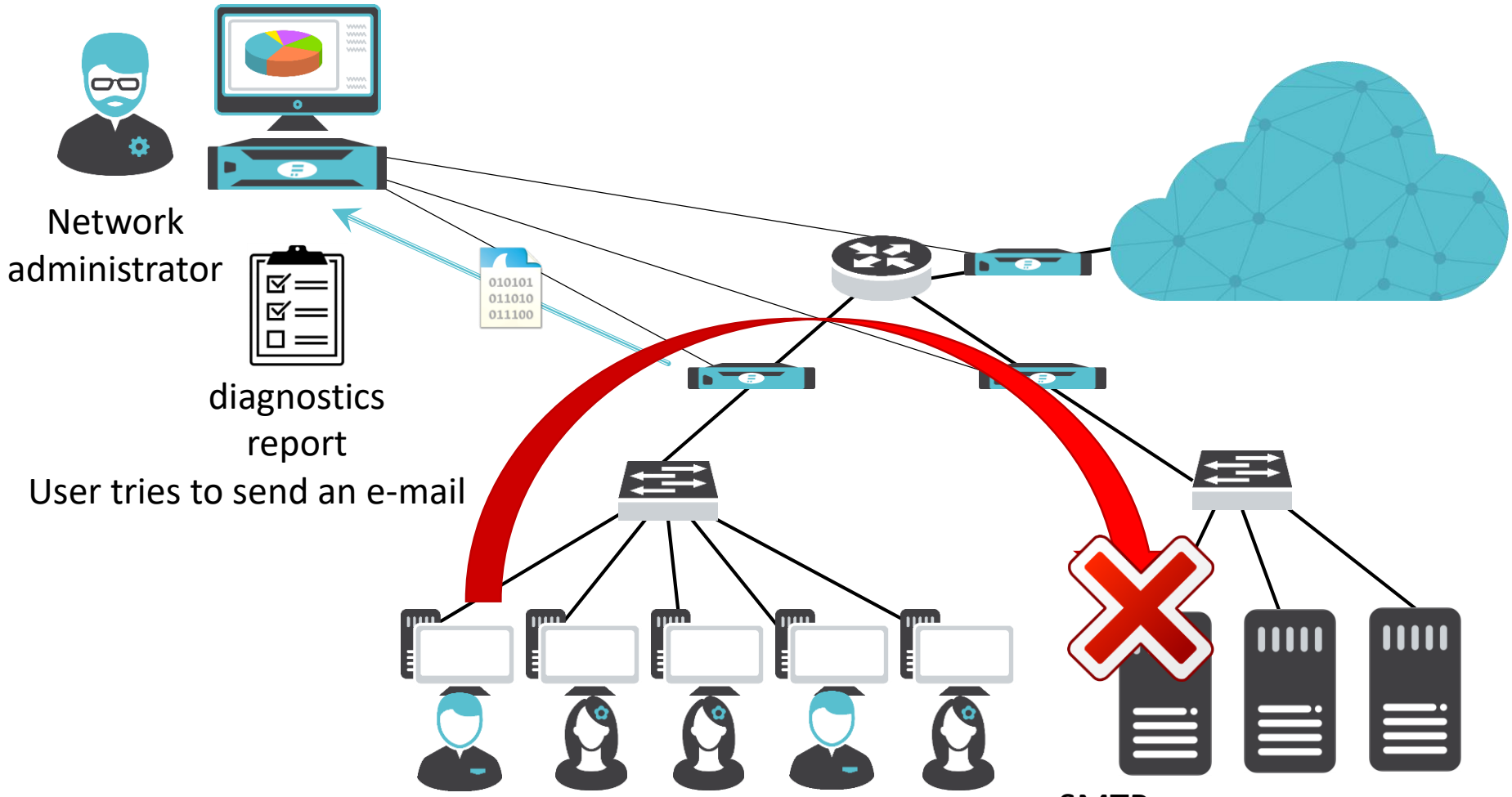
Martin Holkovič, CESNET, Czech Republic
Ondřej Ryšavý, Brno University of Technology, Czech Republic

cesnet

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Motivation

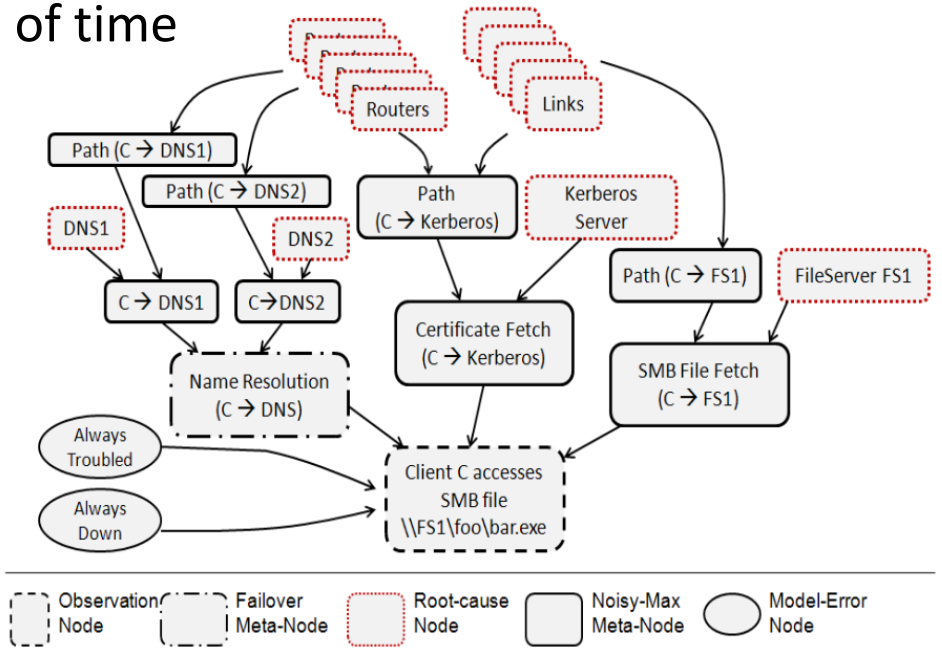
Flowmon
Driving Network Visibility



https://www.flowmon.com/en/products/flowmon/traffic-recorder

Why it is not an easy problem

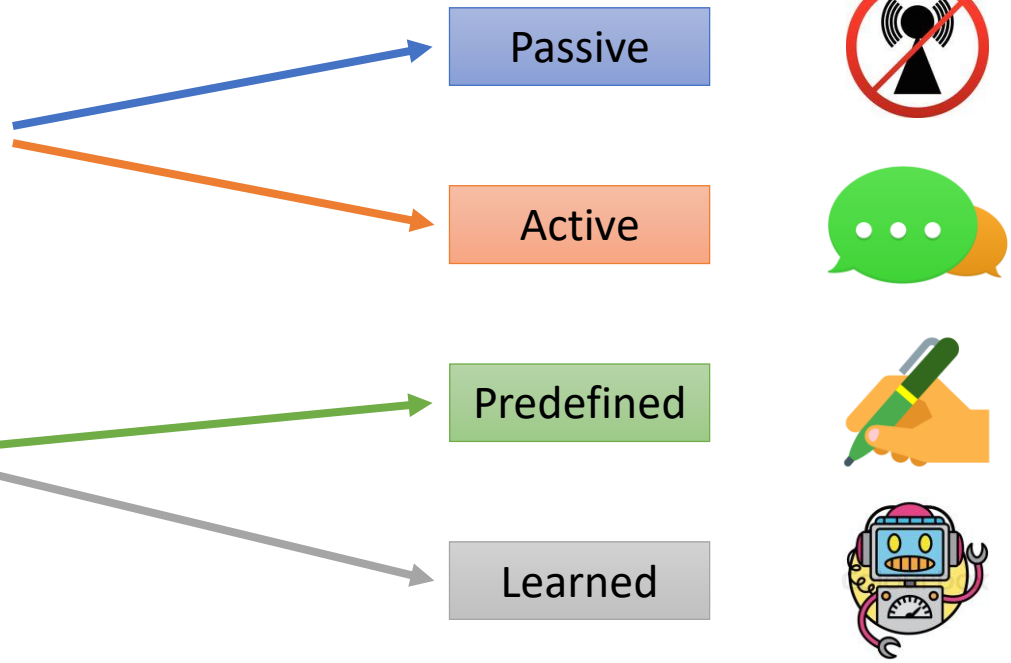
- Each protocol is different
- Each network is different
- Dependencies between services
- Requiring deep knowledge and lot of time



Bahl, P.; Chandra, R.; Greenberg, A.; aj.: Towards highly reliable enterprise network services via inference of multi-level dependencies. In *ACM SIGCOMM Computer CommunicationReview*, ročník 37, ACM, 2007, s. 13–24

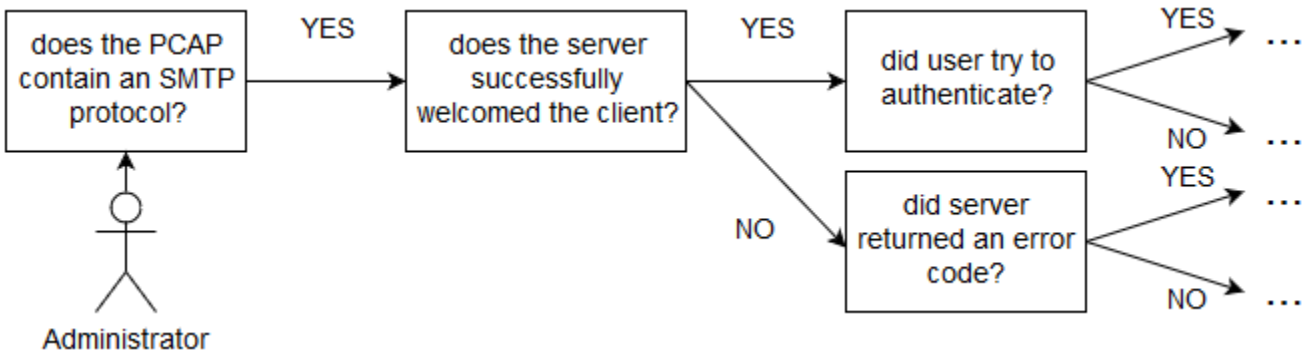
Possible methods

- Wireshark - manual
- How are the data accessed?
- How is the model created?

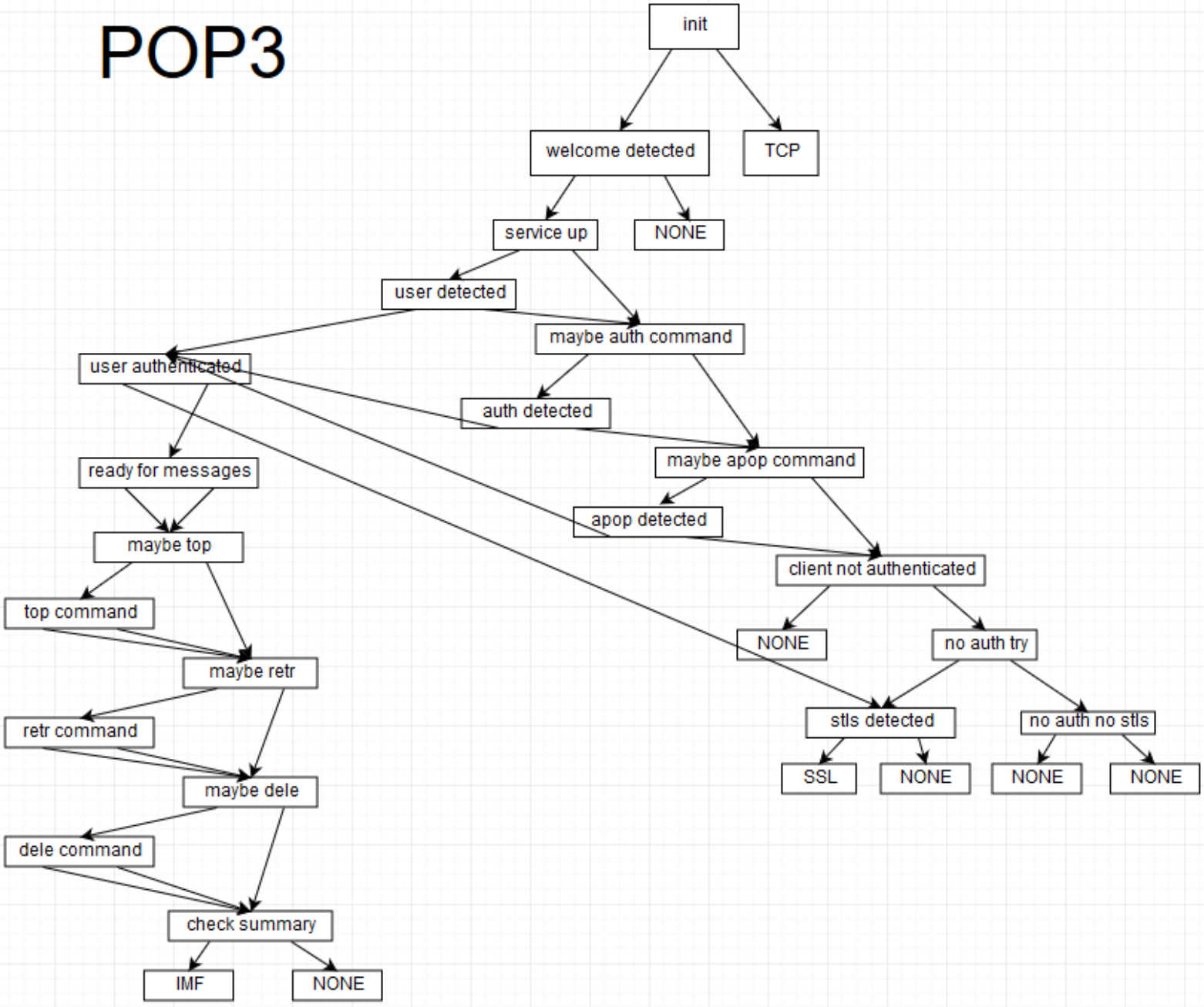


Our goals

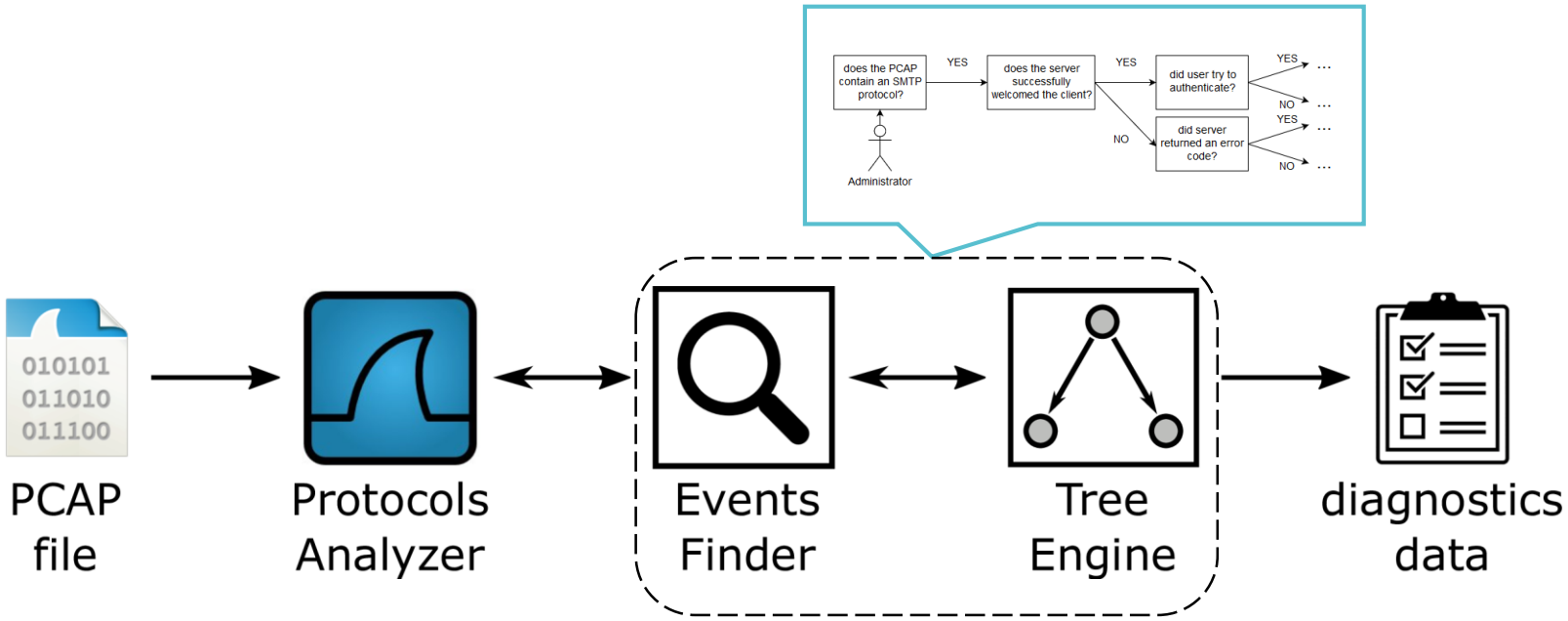
- Passive analysis from PCAP file
- Predefined rule-based tree model
- Automate administrator's actions
- Good-readable diagnostic output
- Easily extendible by an administrator



POP3


init
welcome detected
TCP
service up
NONE
user detected
maybe auth command
user authenticated
auth detected
ready for messages
maybe apop command
apop detected
maybe top
client not authenticated
top command
NONE
no auth try
maybe retr
stls detected
no auth no stls
retr command
SSL
NONE
NONE
NONE
maybe dele
dele command
check summary
IMF
NONE

Proposed architecture

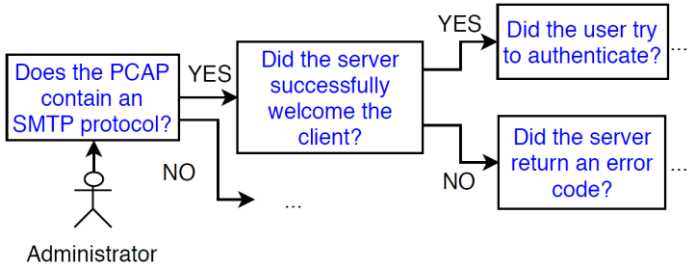


Protocols Analyzer

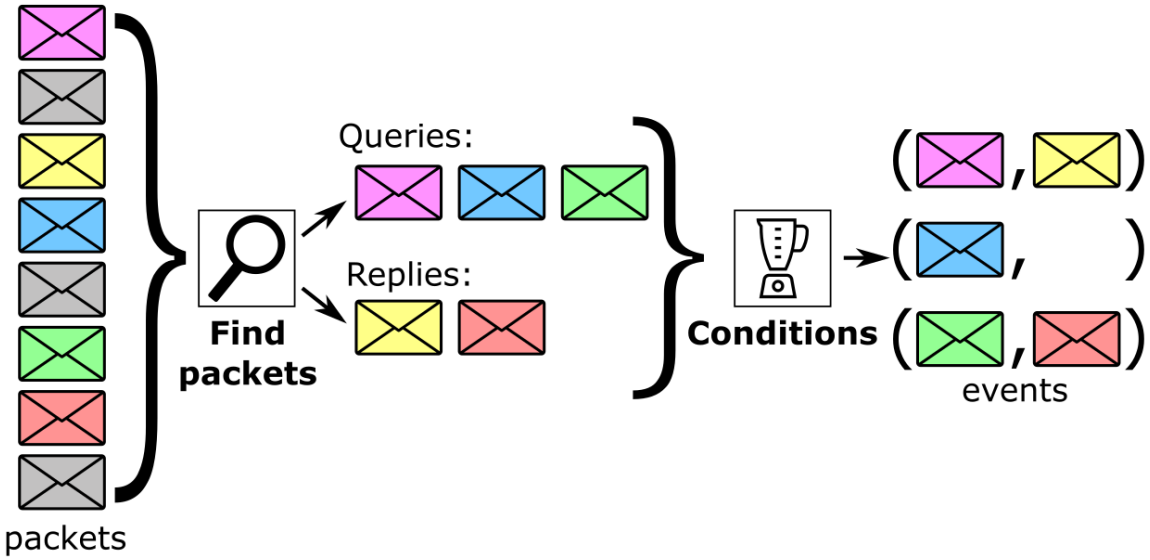
- Using Tshark (Wireshark)
- Support over 3000 protocols and over 227000 fields
- Integrated lower layers analysis
- JSON output

```
...
"eth": {
  "eth.dst": "f0:79:59:72:7c:30",
  "eth.type": "0x00000800",
  ...
},
...
"dns": {
  "dns.id": "0x00007956",
  "dns.flags.response": "0",
  "dns.flags.opcode": "0",
  "dns.qry.name": "mail.patriots.in",
  ...
},
...
```

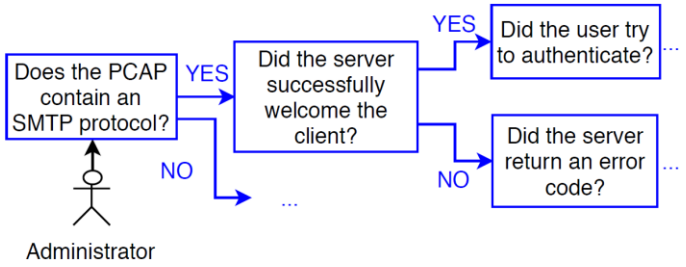
Events Finder



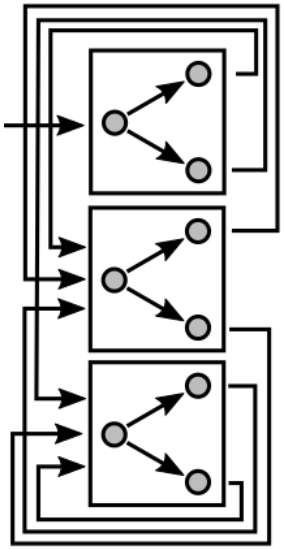
- Simulates questions of a real administrator
  - E.g., SMTP authentication
- Two step process:
  1. Find specific packets
  2. Create tuples from packets fulfilling conditions

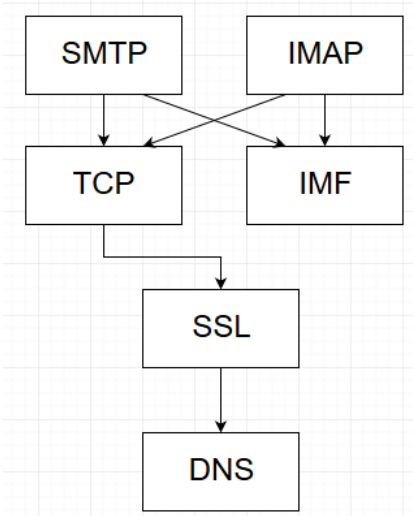


Tree Engine

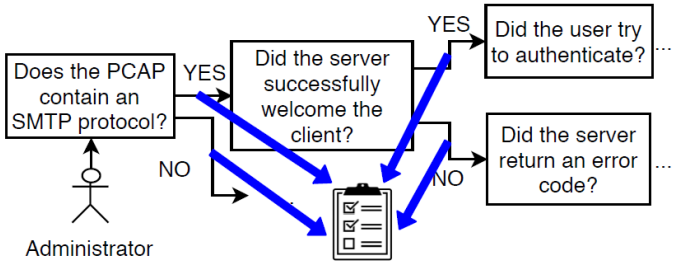


- Binary tree
  - Two next states
- Each node refers to the Events Finder
- State represents the knowledge
- Integrates Python code



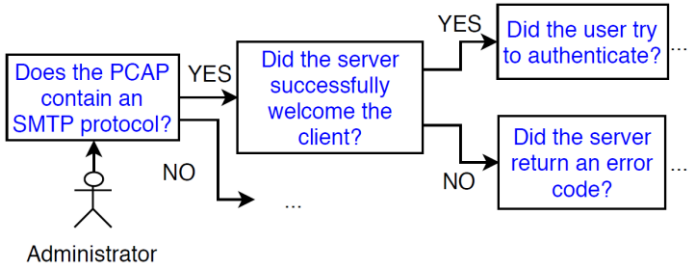


Output creator



- Predefined output records
- Creates links between records
- JSON format

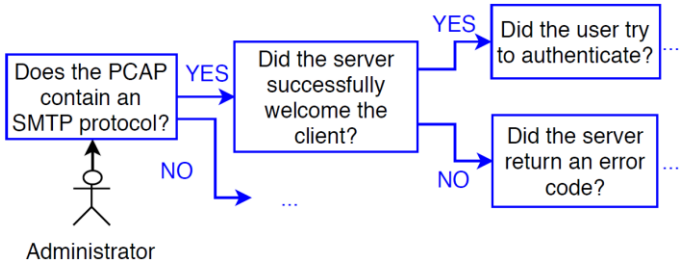
Rules – Events Finder




```
id: RULE_NAME
facts:
    - FACT_NAME_1: FACT_FILTER_1
    - ...
    - FACT_NAME_N: FACT_FILTER_N
params:
    - PARAM_NAME_1
    - ...
    - PARAM_NAME_N
asserts:
    - CONDITION_1
    - ...
    - CONDITION_N
```

```
1  id: welcome ok? # name of the rule
2  facts: # which packets we are looking for
3    - command: smtp.req.command in {"HELO" "EHLO"}
4    - reply: smtp.response.code == "250"
5  asserts: # packets relation constrain
6   - command[tcp.stream] == reply[tcp.stream]
7   - command[tcp.ack] == reply[tcp.seq]
```

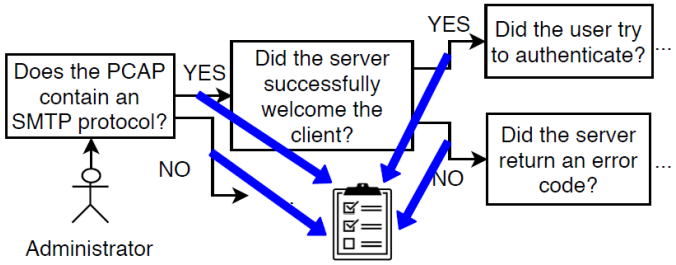
Rules – Tree Engine




```
id: NAME
query: EVENTS_FINDER_RULE
success:
    code: |
        PYTHON_CODE
    state: NEXT_PROTOCOL/NEXT_STATE
fail:
    code: |
        PYTHON_CODE
    state: NEXT_PROTOCOL/NEXT_STATE
```

```
1  id: smtp detected # name of the rule
2  query: welcome ok? # Events Finder rule
3  success:
4    state: client welcomed # next state
5    code: | # Python code follows
6      event("client_welcomed")
7  fail:
8    state: check error # next state
9    code: | # Python code follows
10     event("client_not_welcomed")
```

Rules - Output

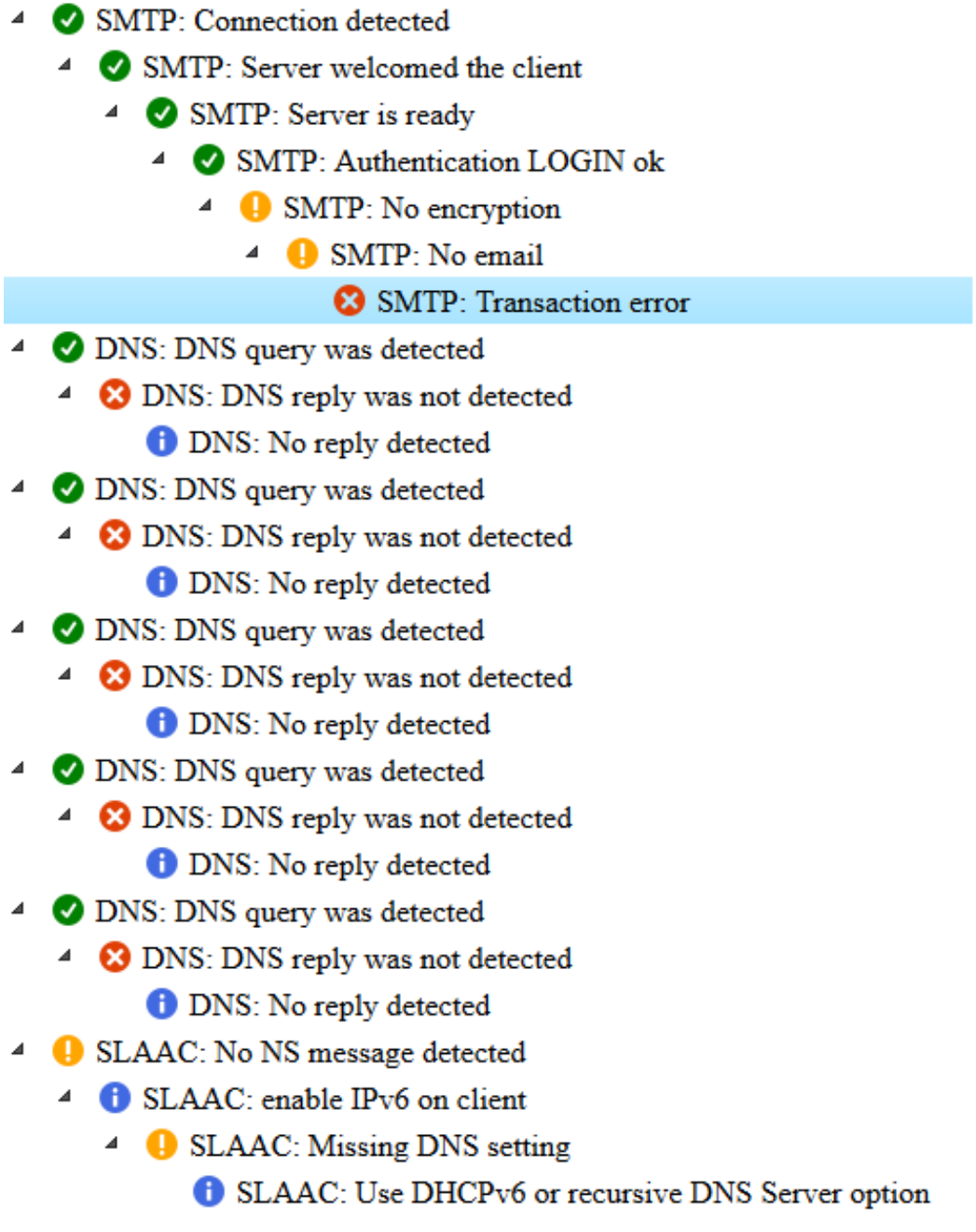



```
id: OUTPUT_RECORD_NAME
description: DESCRIPTION
severity: 'error', 'warning', 'notice', 'information'
message: STRING_WITH_PLACEHOLDERS
fields:
  - name: FIELD_NAME
    description: FIELD_DESCRIPTION
```

```
1 - event:
2     id:          client_welcomed
3     description:  "Server welcomed the client"
4     severity:     information
5     message:      "SMTP server welcomed the client - SMTP service is running."
```

Supported protocols

Protocol	Tree rules	Event rules	Diag. report		
			Success	Warning	Error
DHCP	25	23	10	9	4
DNS	12	12	8	2	6
FTP	24	10	17	5	6
ICMP	4	2	0	0	4
IMAP	15	8	7	0	11
POP	21	7	8	5	10
SIP	38	22	15	1	8
SLAAC	8	7	1	5	2
SMB	27	25	20	4	5
SMTP	17	13	10	5	9
SSL	1	1	1	0	1
TCP	11	11	0	8	3

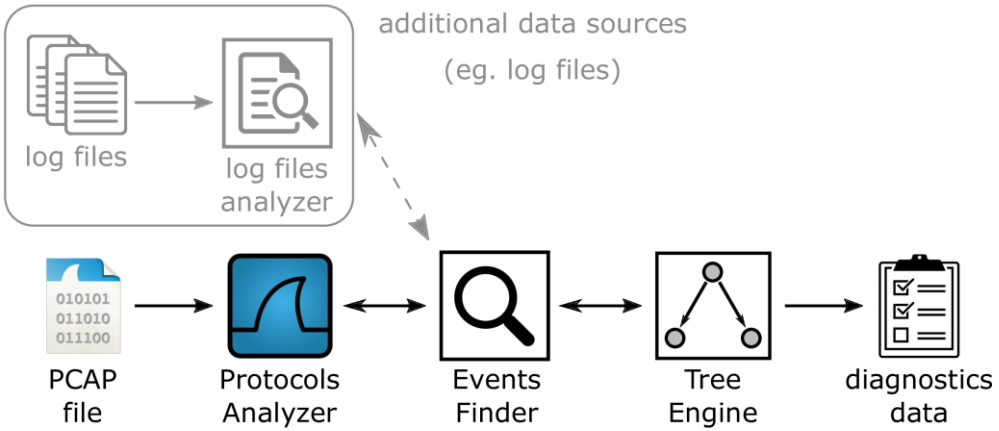


name	some error
description	Transaction error
message	Error code 552 - Requested mail actions aborted - Exceeded storage allocation..
provider	SMTP
severity	error
flow	TCP 74.53.140.153(25)→10.10.1.4(1470)
tcp errors	-
event-record-id	619556f2-02c1-4014-ba0d-41c17eed1885
parent-record-id	8ca78da6-51f1-41c5-98ac-2d6bab2e726f

ip.src	74.53.140.153
ipv6.nxt	
ipv6.src	
frame.time_epoch	1254722769.956765000
frame.number	17
tcp.stream	0
decoded_error	Requested mail actions aborted - Exceeded storage allocation.
udp.srcport	
udp.dstport	
smtp.response.code	552
ipv6.dst	
ip.proto	6
tcp.dstport	1470
udp.stream	
tcp.srcport	25
ip.dst	10.10.1.4
ip.version	4

Future work

- Use another passive data sources
  - Syslog
  - SNMP traps
- Optimize performance
  - Filtering input data
  - Indexing key-data for faster processing



Conclusion

- Network administrators need to diagnose problems
- Diagnostics is time and knowledge requiring activity
- We use PCAP files as the data source
- We have implemented tree-based analysis
- The diagnostic output is good understandable

SMTP: Connection detected
SMTP: Server welcomed the client
SMTP: Server is ready
SMTP: Authentication 'gurpartap@patriots.in' - ok
SMTP: The communication is not encrypted
SMTP: No email has been sent
SMTP: Transaction error code 552 - Requested mail actions aborted - Exceeded storage allocation
SMTP: Empty email account storage (check SPAM folder) or increase the account quota.