Anomaly Detection Based on Simplicity Theory

Giacomo Casoni
Mar Badias Simó

Research Project 1 - #43
Supervisor: Giovanni Sileno
Lecturer: Cees de Laat

TABLE OF CONTENTS

TABLE OF CONTENTS

Simplicity Theory

Calculates **unexpectedness** of a situation ➤ **U(s)**

- **Cognitive probability** in terms of complexity and simplicity, rather than standard mathematical, set-based, terms.

Simplicity Theory

Calculates **unexpectedness** of a situation ➤ **U(s)**

- **Cognitive probability** in terms of complexity and simplicity, rather than standard mathematical, set-based, terms.

- Generation complexity ➤ $C_w(s)$

- Description complexity ➤ $C_d(s)$

Simplicity Theory

Calculates **unexpectedness** of a situation ➤ **U(s)**

- **Cognitive probability** in terms of complexity and simplicity, rather than standard mathematical, set-based, terms.
- Generation complexity ➤ $C_w(s)$
- Description complexity ➤ $C_d(s)$

$$U(s) = C_w(s) - C_d(s)$$

Simplicity Theory

An example

- Fair lottery draw: **1-2-3-4-5-6**
- Same chances than any other combination
- **Odd from a human point of view**

Simplicity Theory

An example

- Fair lottery draw: **1-2-3-4-5-6**
- Same chances than any other combination
- **Odd from a human point of view**

- Same generation cost of other combinations
- Low description cost ("1 to 6")
- Therefore:

$$\uparrow U(s) = C_w(s) - \downarrow C_d(s)$$

Simplicity Theory

A situation is unexpected, in the eyes of an observer, when it is **hard to generate** (high $C_w(s)$) and/or **easy to describe** (low $C_d(s)$).

Anomaly Detection

Anomaly detection systems model the normal behavior of a target system and report abnormal activities, which are analyzed as a possible intrusions.

Research Questions

1. How can an anomaly detection tool based on Simplicity Theory be designed and implemented?

2. How effective said tool can be in detecting anomalies in network logs in a system?

TABLE OF CONTENTS

Putting it Into Practice

$$U(s) = C_w(s) - C_d(s)$$

Putting it Into Practice

$$U(s) = C_w(s) - C_d(s)$$

QUANTIFY COMPLEXITIES

How can generation and description complexity be quantified?

The quantification needs to be **representative** and **comparable**.

Putting it Into Practice

$$U(s) = C_w(s) - C_d(s)$$

SET A CONTEXT

Simplicity Theory allows for observer **point-of-view bias**.

Different observer might have different concepts of “abnormal”.

Set a Context (1)

Define **object prototypes.**

Prototypes, in the conceptual space, are used as baseline to compute generation and description complexity of a given state.

Defined in *n* dimensions, where *n* is the number of features

Set a Context (2)

In our case, one of the categorical features...

Set a Context (2)

In our case, one of the categorical features...

- Source IP: monitor an IP address traffic for abnormal behaviours. (Compromised machine)

Set a Context (2)

In our case, one of the categorical features...

- Source IP: monitor an IP address traffic for abnormal behaviours. (Compromised machine)
- Destinatination IP: monitor for unusual traffic to a specific machine. (Server under attack)

Set a Context (2)

In our case, one of the categorical features...

- Source IP: monitor an IP address traffic for abnormal behaviours. (Compromised machine)
- Destinatination IP: monitor for unusual traffic to a specific machine. (Server under attack)
- Protocol: monitor for abnormal protocol-specific traffic. (Specific attacks)

Set a Context (2)

In our case, one of the categorical features...

- Source IP: monitor an IP address traffic for abnormal behaviours. (Compromised machine)
- Destinatination IP: monitor for unusual traffic to a specific machine. (Server under attack)
- Protocol: monitor for abnormal protocol-specific traffic. (Specific attacks)

...however not necessary

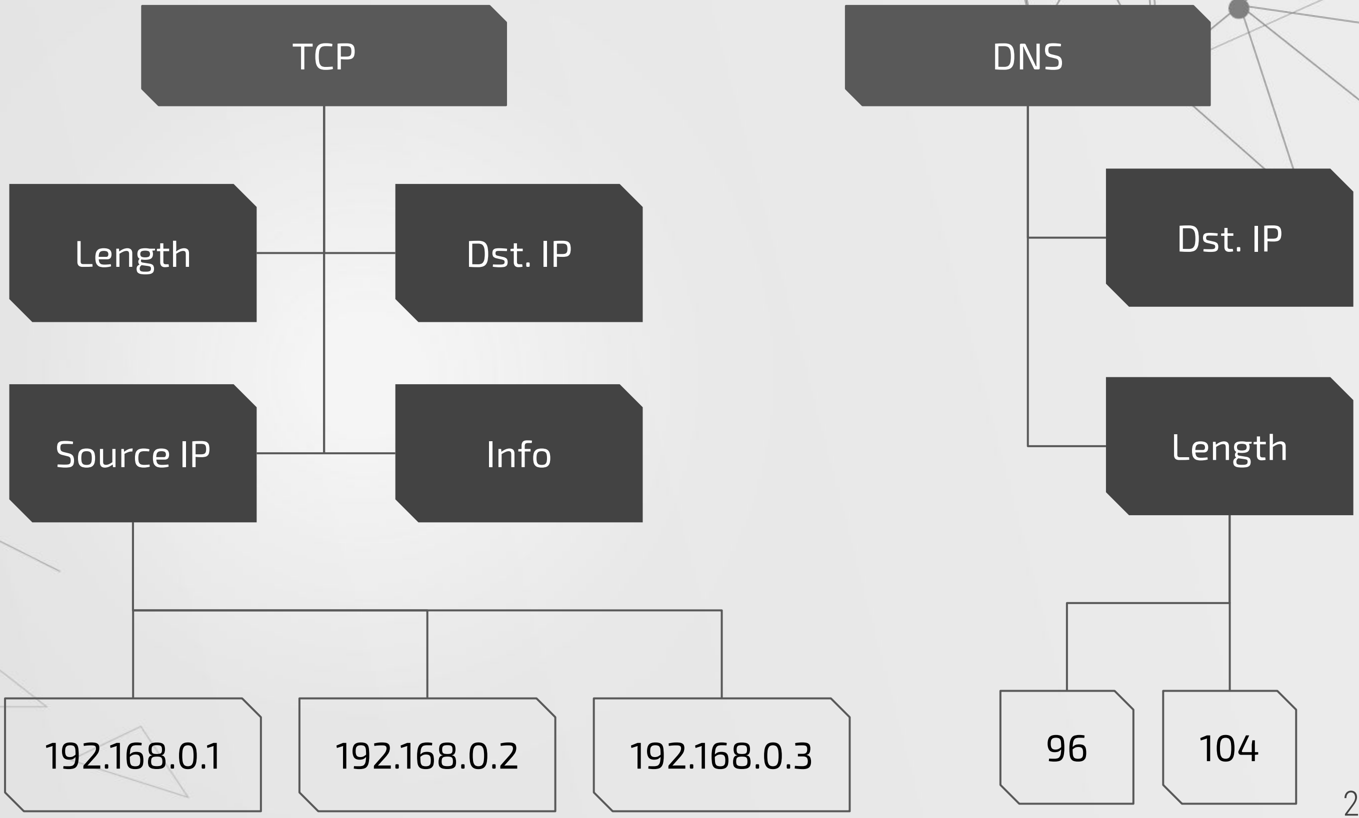
- Combination of categorical features
- K-Prototypes
- No prototypes (aka one prototype)

Set a Context (3)

Object prototypes

Dimensions

Feature prototypes



Quantifying Complexities - Generation (1)

Quantifying Complexities - Generation (1)

"The length of the shortest program that a given environment must execute to achieve a given state"

Quantifying Complexities - Generation (1)

"The length of the shortest program that a given environment must execute to achieve a given state"

Real-life events are often **NOT** like fair lottery, some events are more likely to happen than others ...

Quantifying Complexities - Generation (1)

"The length of the shortest program that a given environment must execute to achieve a given state"

Real-life events are often **NOT** like fair lottery, some events are more likely to happen than others ...

… a ranking of most frequently occurring feature prototypes has to be created.

Quantifying Complexities - Generation (2)

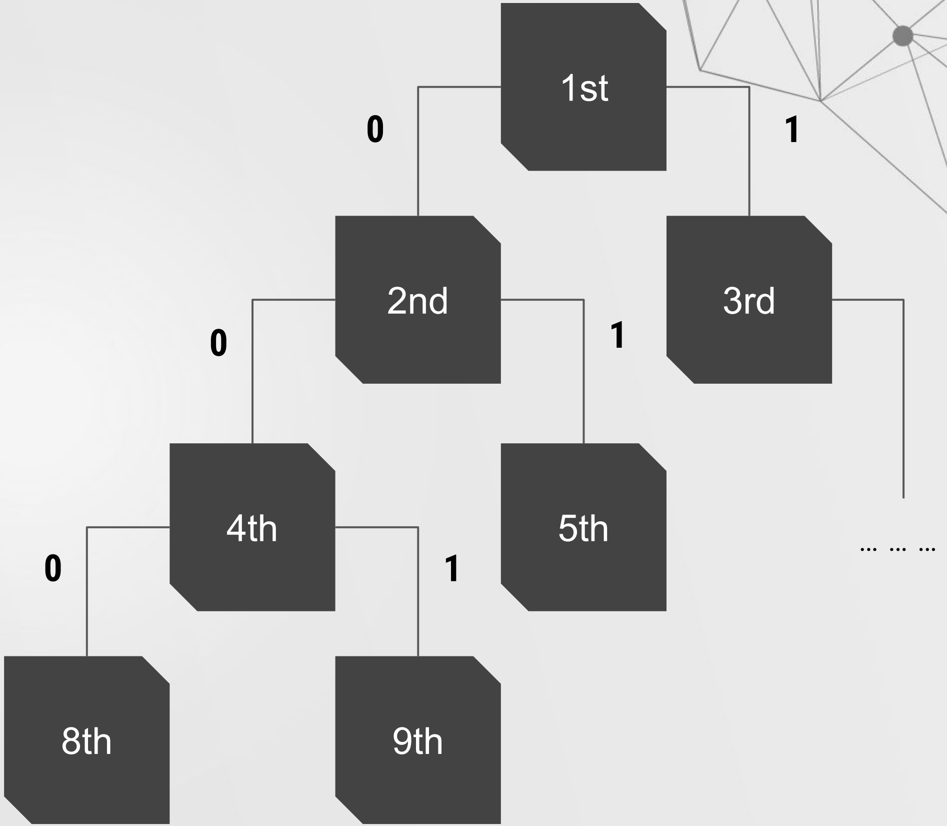
	CODE	COMPLEXITY
1st		0
2nd	0	1
3rd	1	1
4th	00	2
5th	01	2
6th	10	2
7th	11	2
8th	000	3
9th	001	3

Quantifying Complexities - Generation (2)

	CODE	COMPLEXITY
192.168.0.1		0
192.168.0.2	0	1
192.168.0.3	1	1
192.168.0.4	00	2
192.168.0.5	01	2
192.168.0.6	10	2
192.168.0.7	11	2
192.168.0.8	000	3
192.168.0.9	001	3

Quantifying Complexities - Generation (2)

	CODE	COMPLEXITY
192.168.0.1		0
192.168.0.2	0	1
192.168.0.3	1	1
192.168.0.4	00	2
192.168.0.5	01	2
192.168.0.6	10	2
192.168.0.7	11	2
192.168.0.8	000	3
192.168.0.9	001	3



Quantifying Complexities - Description (1)

Quantifying Complexities - Description (1)

”The shortest possible description of a state that an observer can produce to discriminate it without ambiguity”

Quantifying Complexities - Description (1)

”The shortest possible description of a state that an observer can produce to discriminate it without ambiguity”

It could be the same as the generation complexity...

Quantifying Complexities - Description (1)

"The shortest possible description of a state that an observer can produce to discriminate it without ambiguity"

It could be the same as the generation complexity...

… but an observer can also use its own memory to achieve simpler descriptions.

Quantifying Complexities - Description (1)

"The shortest possible description of a state that an observer can produce to discriminate it without ambiguity"

It could be the same as the generation complexity...

… but an observer can also use its own memory to achieve simpler descriptions.

The **cheapest** option is chosen.

Quantifying Complexities - Description (2)

At observation time N, the stack pointer is here.

	MOVES	COMPLEXITY
N-1	0	1
N-2	1	1
N-3	2 (10)	2
N-4	3 (11)	2
N-5	4 (100)	3
N-6	5 (101)	3
N-7	6 (110)	3
N-8	7 (111)	3
N-9	8 (1000)	4

Quantifying Complexities - Numerical (1)

PROBLEM!

Previous methods work for categorical feature prototypes.
Numerical feature prototypes cannot be ranked.

Quantifying Complexities - Numerical (1)

PROBLEM!

Previous methods work for categorical feature prototypes.
Numerical feature prototypes cannot be ranked.

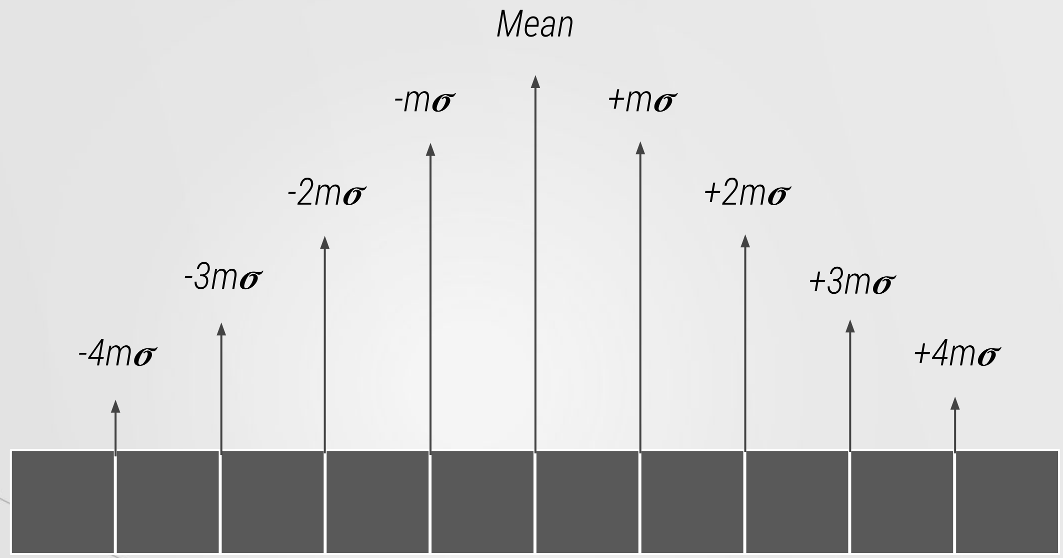
Idea: numerical feature prototypes could be transformed into categorical ones.

Quantifying Complexities - Numerical (2)

SOLUTION - Binary Tree

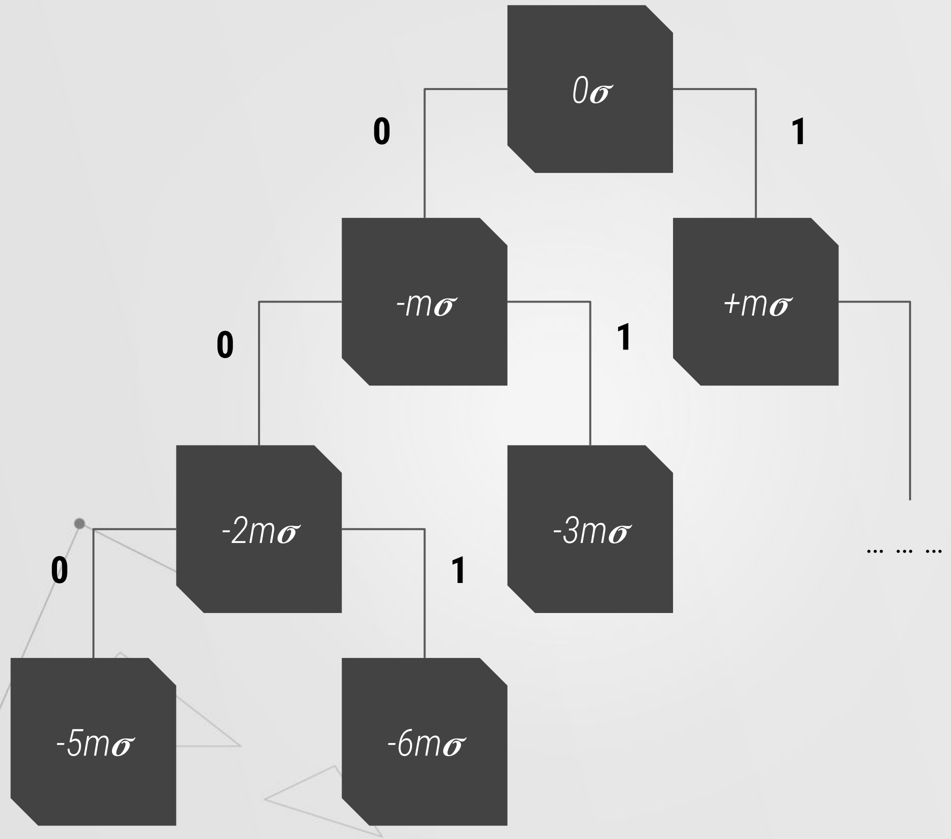
Compute mean and standard deviation over all the possible feature prototypes.
Describe a feature prototype as being $n * (m\sigma)$ away from the mean.
Populate the tree with $m\sigma$ intervals, starting from the closest to the mean.

Quantifying Complexities - Numerical (3)



000	01	00	0			1	10	11	111	CODES
3	2	2	1	0	0	1	2	2	3	COMPLEXITIES

Quantifying Complexities - Numerical (4)



Quantifying Complexities - Numerical (5)

SOLUTION - Memory Stack

Compute mean and standard deviation over all the possible feature prototypes.
Describe an observation as being $n * (m\sigma)$ away from a previous observation.
Complexity is given by the depth of the previous observation **and** its distance from the current observation.

Quantifying Complexities - Numerical (6)

At observation time (N, d) the stack pointer is here.

	MOVES	COMPLEXITY
(N-1, d_1)	0	1+log(d-d_1)
(N-2, d_2)	1	1+log(d-d_2)
(N-3, d_3)	2 (10)	2+log(d-d_3)
(N-4, d_4)	3 (11)	2+log(d-d_4)
(N-5, d_5)	4 (100)	3+log(d-d_5)
(N-6, d_6)	5 (101)	3+log(d-d_6)
(N-7, d_7)	6 (110)	3+log(d-d_7)
(N-8, d_8)	7 (111)	3+log(d-d_8)
(N-9, d_9)	8 (1000)	4+log(d-d_9)

TABLE OF CONTENTS

Dataset transformation

DARPA 1999 IDS dataset

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	HewlettP_61:aa:c9	HewlettP_61:aa:c9	LLC	54	U P, func=TEST; DSAP NULL LSAP Individual, SSAP NetBIOS Command
2	0.603594	Cisco_38:46:32	Cisco_38:46:32	LOOP	60	Reply
3	0.703093	172.16.112.20	192.168.1.10	DNS	78	Standard query 0x067c A jupiter.cherry.org
4	0.704269	192.168.1.10	172.16.112.20	DNS	134	Standard query response 0x067c A jupiter.cherry.org A 196.37.75.158 NS jupiter.cherry.org A 196.37.75.158
5	0.713216	172.16.112.194	196.37.75.158	TCP	60	1024 → 25 [SYN] Seq=0 Win=512 Len=0 MSS=1460
6	0.713563	196.37.75.158	172.16.112.194	TCP	60	25 → 1024 [SYN, ACK] Seq=0 Ack=1 Win=32736 Len=0 MSS=1460
7	0.716372	172.16.112.194	196.37.75.158	TCP	60	1024 → 25 [ACK] Seq=1 Ack=1 Win=32120 Len=0
8	0.880191	192.168.1.10	172.16.112.20	DNS	87	Standard query 0x577f PTR 194.112.16.172.in-addr.arpa
9	0.881494	172.16.112.20	192.168.1.10	DNS	176	Standard query response 0x577f PTR 194.112.16.172.in-addr.arpa PTR falcon.eyrie.af.mil NS hobbes.eyrie.af.mil A 172.16.1
10	0.882980	192.168.1.10	172.16.112.20	DNS	79	Standard query 0x5780 A falcon.eyrie.af.mil
11	0.884051	172.16.112.20	192.168.1.10	DNS	144	Standard query response 0x5780 A falcon.eyrie.af.mil A 172.16.112.194 NS hobbes.eyrie.af.mil A 172.16.112.20
12	0.969062	196.37.75.158	172.16.112.194	SMTP	140	S: 220 jupiter.cherry.org Sendmail 4.1/SMI-4.1 ready at Mon, 29 Mar 1999 08:00:04 -0500
13	0.982806	172.16.112.194	196.37.75.158	TCP	60	1024 → 25 [ACK] Seq=1 Ack=87 Win=32120 Len=0
14	1.011997	172.16.112.194	196.37.75.158	SMTP	80	C: EHLO falcon.eyrie.af.mil
15	1.012229	196.37.75.158	172.16.112.194	SMTP	80	S: 500 Command unrecognized
16	1.013261	172.16.112.194	196.37.75.158	SMTP	80	C: HELO falcon.eyrie.af.mil
17	1.013500	196.37.75.158	172.16.112.194	SMTP	102	S: 250 (falcon.eyrie.af.mil) pleased to meet you.
18	1.014378	172.16.112.194	196.37.75.158	SMTP	96	C: MAIL From:<wardelld@falcon.eyrie.af.mil>
19	1.014625	196.37.75.158	172.16.112.194	SMTP	103	S: 250 <wardelld@falcon.eyrie.af.mil>... Sender Ok
20	1.015585	172.16.112.194	196.37.75.158	SMTP	93	C: RCPT To:<phyllisn@jupiter.cherry.org>
21	1.015820	196.37.75.158	172.16.112.194	SMTP	92	S: 250 <phyllisn@jupiter.cherry.org> OK
22	1.016638	172.16.112.194	196.37.75.158	SMTP	60	C: DATA
23	1.017158	196.37.75.158	172.16.112.194	SMTP	104	S: 354 Enter mail, end with "." on a line by itself
24	1.019570	172.16.112.194	196.37.75.158	SMTP|I…	1018	subject: Neural net, such as an end end, , Neural net, such as an end end of items, from; isn't as a putative hit , at
25	1.020421	196.37.75.158	172.16.112.194	SMTP	73	S: 250 Mail accepted
26	1.021169	172.16.112.194	196.37.75.158	SMTP	60	C: QUIT
27	1.021428	196.37.75.158	172.16.112.194	SMTP	78	S: 221 Closing connection
28	1.022016	196.37.75.158	172.16.112.194	TCP	60	25 → 1024 [FIN, ACK] Seq=341 Ack=1110 Win=32736 Len=0
29	1.022454	172.16.112.194	196.37.75.158	TCP	60	1024 → 25 [ACK] Seq=1110 Ack=342 Win=32120 Len=0

Dataset transformation

DARPA 1999 IDS dataset

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	HewlettP_61:aa:c9	HewlettP_61:aa:c9	LLC	54	U P, func=TEST; DSAP NULL LSAP Individual, SSAP NetBIOS Command
2	0.603594	Cisco_38:46:32	Cisco_38:46:32	LOOP	60	Reply
3	0.703093	172.16.112.20	192.168.1.10	DNS	78	Standard query 0x067c A jupiter.cherry.org
4	0.704269	192.168.1.10	172.16.112.20	DNS	134	Standard query response 0x067c A jupiter.cherry.org A 196.37.75.158 NS jupiter.cherry.org A 196.37.75.158
5	0.713216	172.16.112.194	196.37.75.158	TCP	60	1024 → 25 [SYN] Seq=0 Win=512 Len=0 MSS=1460
6	0.713563	196.37.75.158	172.16.112.194	TCP	60	25 → 1024 [SYN, ACK] Seq=0 Ack=1 Win=32736 Len=0 MSS=1460
7	0.716372	172.16.112.194	196.37.75.158	TCP	60	1024 → 25 [ACK] Seq=1 Ack=1 Win=32120 Len=0
8	0.880191	192.168.1.10	172.16.112.20	DNS	87	Standard query 0x577f PTR 194.112.16.172.in-addr.arpa
9	0.881494	172.16.112.20	192.168.1.10	DNS	176	Standard query response 0x577f PTR 194.112.16.172.in-addr.arpa PTR falcon.eyrie.af.mil NS hobbes.eyrie.af.mil A 172.16.1
10	0.882980	192.168.1.10	172.16.112.20	DNS	79	Standard query 0x5780 A falcon.eyrie.af.mil
11	0.884051	172.16.112.20	192.168.1.10	DNS	144	Standard query response 0x5780 A falcon.eyrie.af.mil A 172.16.112.194 NS hobbes.eyrie.af.mil A 172.16.112.20
12	0.969062	196.37.75.158	172.16.112.194	SMTP	140	S: 220 jupiter.cherry.org Sendmail 4.1/SMI-4.1 ready at Mon, 29 Mar 1999 08:00:04 -0500
13	0.982806	172.16.112.194	196.37.75.158	TCP	60	1024 → 25 [ACK] Seq=1 Ack=87 Win=32120 Len=0
14	1.011997	172.16.112.194	196.37.75.158	SMTP	80	C: EHLO falcon.eyrie.af.mil
15	1.012229	196.37.75.158	172.16.112.194	SMTP	80	S: 500 Command unrecognized
16	1.013261	172.16.112.194	196.37.75.158	SMTP	80	C: HELO falcon.eyrie.af.mil
17	1.013500	196.37.75.158	172.16.112.194	SMTP	102	S: 250 (falcon.eyrie.af.mil) pleased to meet you.
18	1.014378	172.16.112.194	196.37.75.158	SMTP	96	C: MAIL From:<wardelld@falcon.eyrie.af.mil>
19	1.014625	196.37.75.158	172.16.112.194	SMTP	103	S: 250 <wardelld@falcon.eyrie.af.mil>... Sender Ok
20	1.015585	172.16.112.194	196.37.75.158	SMTP	93	C: RCPT To:<phyllisn@jupiter.cherry.org>
21	1.015820	196.37.75.158	172.16.112.194	SMTP	92	S: 250 <phyllisn@jupiter.cherry.org> OK
22	1.016638	172.16.112.194	196.37.75.158	SMTP	60	C: DATA
23	1.017158	196.37.75.158	172.16.112.194	SMTP	104	S: 354 Enter mail, end with "." on a line by itself
24	1.019570	172.16.112.194	196.37.75.158	SMTP|I…	1018	subject: Neural net, such as an end end, , Neural net, such as an end end of items, from; isn't as a putative hit , at
25	1.020421	196.37.75.158	172.16.112.194	SMTP	73	S: 250 Mail accepted
26	1.021169	172.16.112.194	196.37.75.158	SMTP	60	C: QUIT
27	1.021428	196.37.75.158	172.16.112.194	SMTP	78	S: 221 Closing connection
28	1.022016	196.37.75.158	172.16.112.194	TCP	60	25 → 1024 [FIN, ACK] Seq=341 Ack=1110 Win=32736 Len=0
29	1.022454	172.16.112.194	196.37.75.158	TCP	60	1024 → 25 [ACK] Seq=1110 Ack=342 Win=32120 Len=0

Create templates for each protocol

Calculate Levenshtein distance

Dataset transformation

DARPA 1999 IDS dataset

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	HewlettP_61:aa:c9	HewlettP_61:aa:c9	LLC	54	U P, func=TEST; DSAP NULL LSAP Individual, SSAP NetBIOS Command
2	0.603594	Cisco_38:46:32	Cisco_38:46:32	LOOP	60	Reply
3	0.703093	172.16.112.20	192.168.1.10	DNS	78	Standard query 0x067c A jupiter.cherry.org
4	0.704269	192.168.1.10	172.16.112.20	DNS	134	Standard query response 0x067c A jupiter.cherry.org A 196.37.75.158 NS jupiter.cherry.org A 196.37.75.158
5	0.713216	172.16.112.194	196.37.75.158	TCP	60	1024 → 25 [SYN] Seq=0 Win=512 Len=0 MSS=1460
6	0.713563	196.37.75.158	172.16.112.194	TCP	60	25 → 1024 [SYN, ACK] Seq=0 Ack=1 Win=32736 Len=0 MSS=1460
7	0.716372	172.16.112.194	196.37.75.158	TCP	60	1024 → 25 [ACK] Seq=1 Ack=1 Win=32120 Len=0
8	0.880191	192.168.1.10	172.16.112.20	DNS	87	Standard query 0x577f PTR 194.112.16.172.in-addr.arpa
9	0.881494	172.16.112.20	192.168.1.10	DNS	176	Standard query response 0x577f PTR 194.112.16.172.in-addr.arpa PTR falcon.eyrie.af.mil NS hobbes.eyrie.af.mil A 172.16.1:
10	0.882980	192.168.1.10	172.16.112.20	DNS	79	Standard q
11	0.884051	172.16.112.20	192.168.1.10	DNS	144	Standard q
12	0.969062	196.37.75.158	172.16.112.194	SMTP	140	S: 220 jup
13	0.982806	172.16.112.194	196.37.75.158	TCP	60	1024 → 25
14	1.011997	172.16.112.194	196.37.75.158	SMTP	80	C: EHLO fa
15	1.012229	196.37.75.158	172.16.112.194	SMTP	80	S: 500 Com
16	1.013261	172.16.112.194	196.37.75.158	SMTP	80	C: HELO fa
17	1.013500	196.37.75.158	172.16.112.194	SMTP	102	S: 250 (fa
18	1.014378	172.16.112.194	196.37.75.158	SMTP	96	C: MAIL Fr
19	1.014625	196.37.75.158	172.16.112.194	SMTP	103	S: 250 <wa
20	1.015585	172.16.112.194	196.37.75.158	SMTP	93	C: RCPT To
21	1.015820	196.37.75.158	172.16.112.194	SMTP	92	S: 250 <ph
22	1.016638	172.16.112.194	196.37.75.158	SMTP	60	C: DATA
23	1.017158	196.37.75.158	172.16.112.194	SMTP	104	S: 354 Ent
24	1.019570	172.16.112.194	196.37.75.158	SMTP|I…	1018	subject: N
25	1.020421	196.37.75.158	172.16.112.194	SMTP	73	S: 250 Mai
26	1.021169	172.16.112.194	196.37.75.158	SMTP	60	C: QUIT
27	1.021428	196.37.75.158	172.16.112.194	SMTP	78	S: 221 Clo
28	1.022016	196.37.75.158	172.16.112.194	TCP	60	25 → 1024
29	1.022454	172.16.112.194	196.37.75.158	TCP	60	1024 → 25

```
1,0.000000,Cisco_38:46:33,Cisco_38:46:33,LOOP,60,2
2,0.096519,172.16.112.20,192.168.1.10,DNS,78,26
3,0.101814,192.168.1.10,172.16.112.20,DNS,134,8
4,0.106695,172.16.112.194,196.37.75.158,TCP,60,28
5,0.111396,196.37.75.158,172.16.112.194,TCP,60,37
6,0.111587,172.16.112.194,196.37.75.158,TCP,60,24
7,0.275928,192.168.1.10,172.16.112.20,DNS,87,35
8,0.276578,172.16.112.20,192.168.1.10,DNS,176,72
9,0.278723,192.168.1.10,172.16.112.20,DNS,79,27
10,0.279158,172.16.112.20,192.168.1.10,DNS,144,49
```

- Converted to CSV
- Info field **templated** and **Levenshtein distance** calculated

Features definition

Log line: "5, 0.111396, 196.37.75.158, 172.16.112.194, TCP, 60, 37"

- **196.37.75.158** Source IP
- **172.16.112.194** Destination IP
- **TCP** Protocol
- **60** Length of the packet
- **37** Information - Levenshtein string distance from the template

TABLE OF CONTENTS

Implementation (1)

- Object protocol are based on Protocols (same could have been done with any other feature)
- Source IP and Destination IP are categorical values
- Length and Info are numerical values

Implementation (1)

- Object protocol are based on Protocols (same could have been done with any other feature)
- Source IP and Destination IP are categorical values
- Length and Info are numerical values

Implementation caveats…

Implementation (1)

- Object protocol are based on Protocols (same could have been done with any other feature)
- Source IP and Destination IP are categorical values
- Length and Info are numerical values

Implementation caveats…

- When a new feature prototype appears (i.e. a new IP address for a protocol), it is added as a leaf to the binary tree.

Implementation (1)

- Object protocol are based on Protocols (same could have been done with any other feature)
- Source IP and Destination IP are categorical values
- Length and Info are numerical values

Implementation caveats…

- When a new feature prototype appears (i.e. a new IP address for a protocol), it is added as a leaf to the binary tree.
- When a new object prototype appears (i.e. a new protocol), no action is taken, other than generating a message.

Implementation (2)

Feature prototypes definitions are generated separately for categorical and numerical dimensions.

- Numerical feature prototype definitions contain the mean and the standard deviation for a given dimension.
- Categorical feature prototype definitions contain the ranking of the feature prototypes for a given dimension.

CATEGORICAL


```
"CDP": {
  "sources": {
    "Cisco_38:46:33": 12785,
    "Cisco_38:46:32": 11465
  },
  "destinations": {
    "CDP/VTP/DTP/PAgP/UDLD": 24250
  },
  "sources_ranking": [
    "Cisco_38:46:33",
    "Cisco_38:46:32"
  ],
  "destinations_ranking": [
    "CDP/VTP/DTP/PAgP/UDLD"
  ]
},
```


NUMERICAL


```
"TCPCL": {
  "length": {
    "entries": [
      63,
      89,
      63,
      89
    ],
    "mean": 76,
    "stdev": 15.01110699893027
  },
  "variables": {
    "entries": [
      0,
      0,
      0,
      0
    ],
    "mean": 0,
    "stdev": 0.0
  }
}
```

TABLE OF CONTENTS

Testing and Results (1)

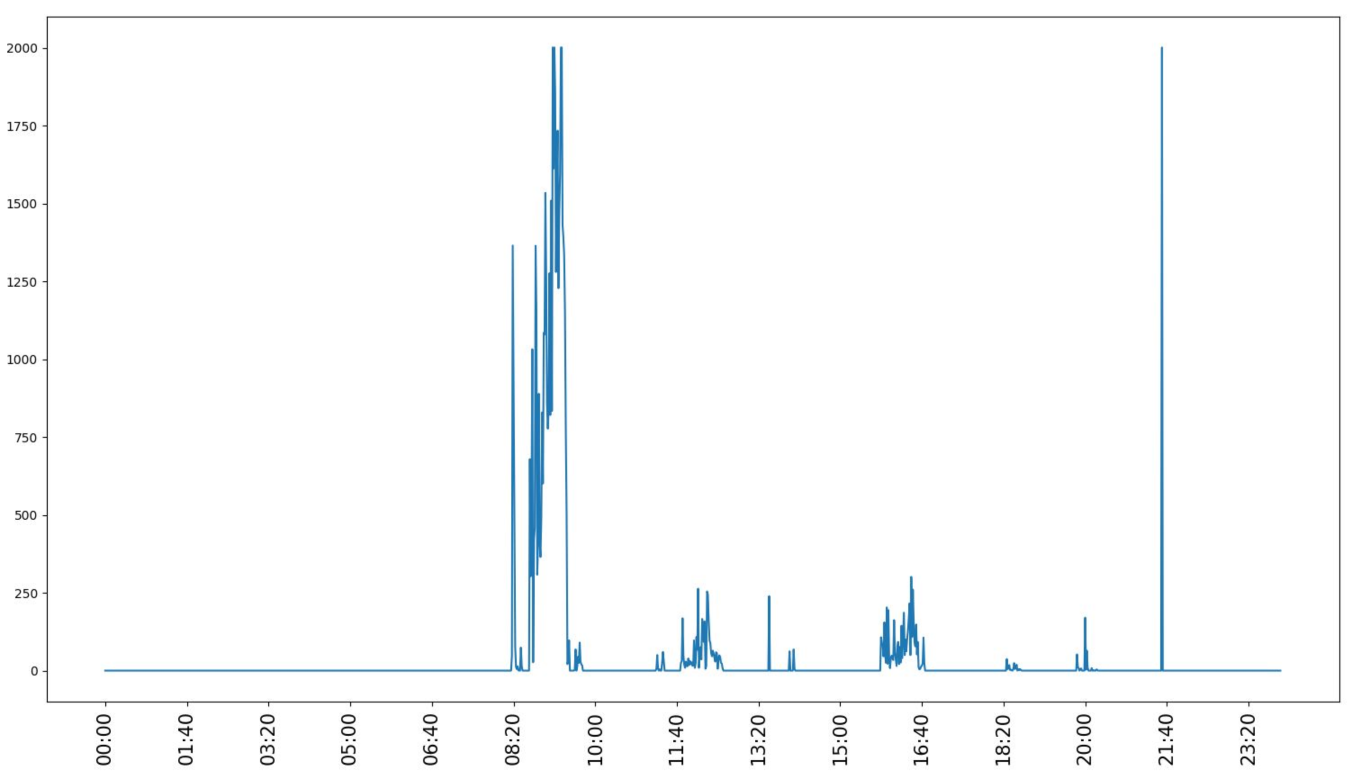
- Training done over weeks 1 and 3.
- Testing done on week 4.
- Testing carried out only on inside captures.

Testing and Results (1)

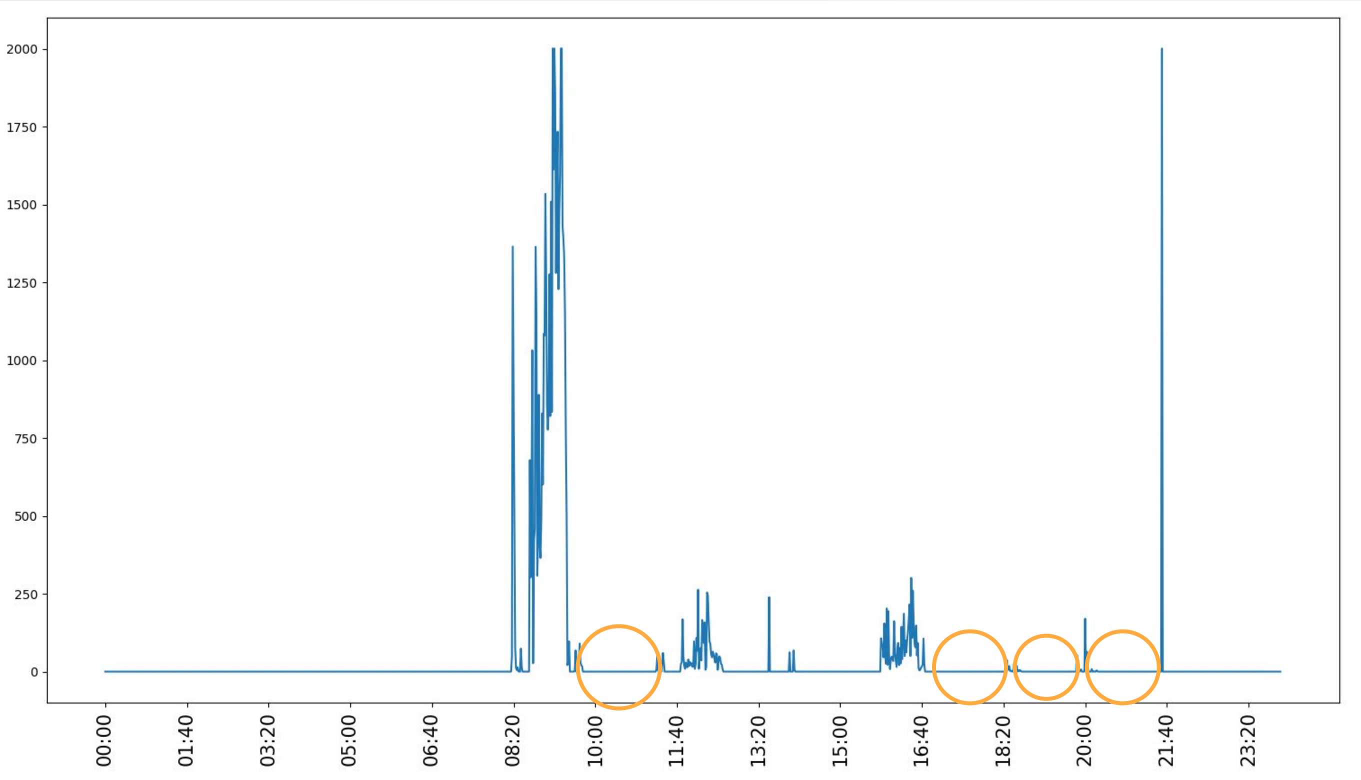
- Training done over weeks 1 and 3.
- Testing done on week 4.
- Testing carried out only on inside captures.

- 96.4% attacks detected (accuracy)
- 80.6% true positives (= 0.81 precision)

Testing and Results (2)

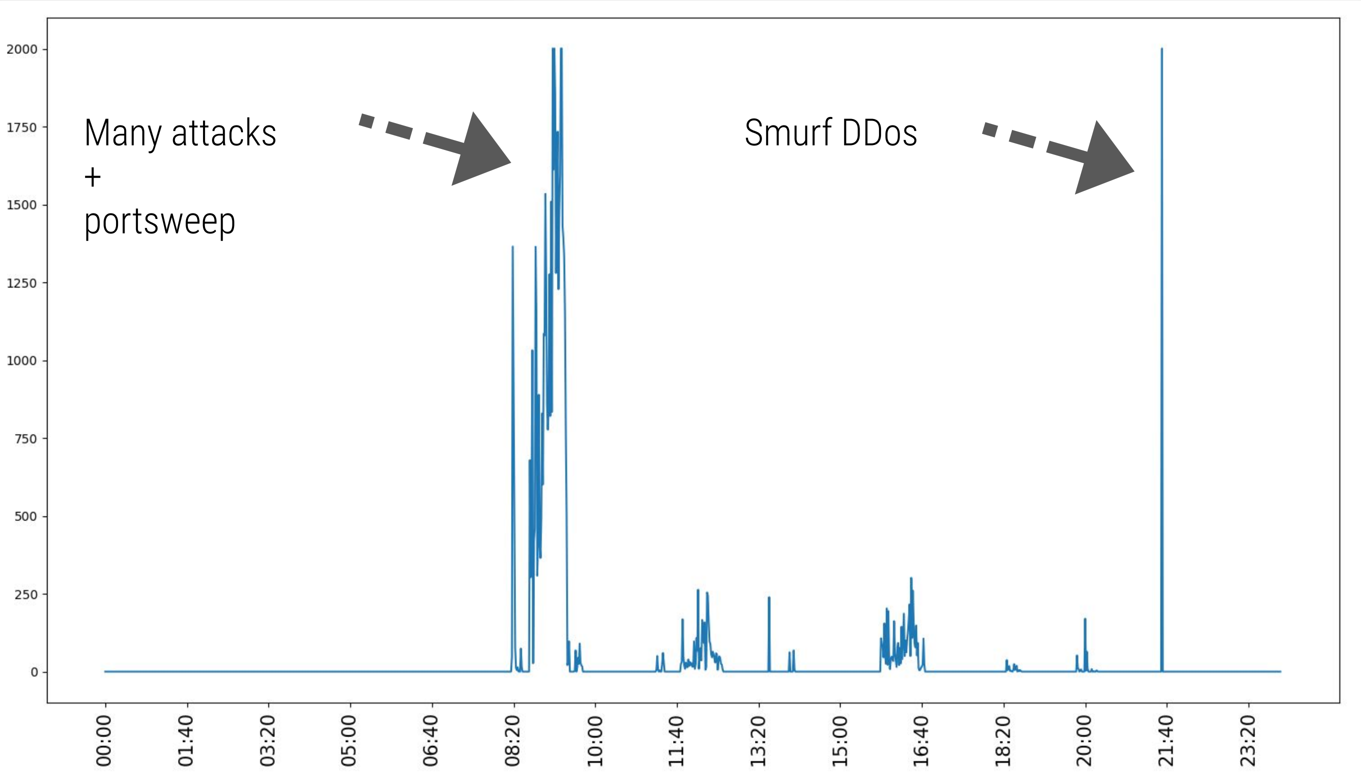


Testing and Results (2)



- From 09:39 to 11:15
- From 16:32 to 18:24
- From 18:27 to 19:50
- From 20:03 to 21:34

Testing and Results (2)



Conclusions (1)

1. How can an anomaly detection tool based on Simplicity Theory be designed and implemented?

2. How effective said tool can be in detecting anomalies in network logs in a system?

Conclusions (1)

1. How can an anomaly detection tool based on Simplicity Theory be designed and implemented?

2. **How effective said tool can be in detecting anomalies in network logs in a system?**

Conclusions (2)

- *“Anomalous Payload-based Network Intrusion Detection”,* Ke Wang, Salvatore J. Stolfo
- *“Robust Support Vector Machines for Anomaly Detection in Computer Security”,* Wenjie Hu et al.
- *“Hierarchical Kohonenen Net for Anomaly Detection in Network Security”,* Suseela T. Sarasamm et al.

Usual false positives rates between <1% and 3%
Accuracy usually between 90% and 94%

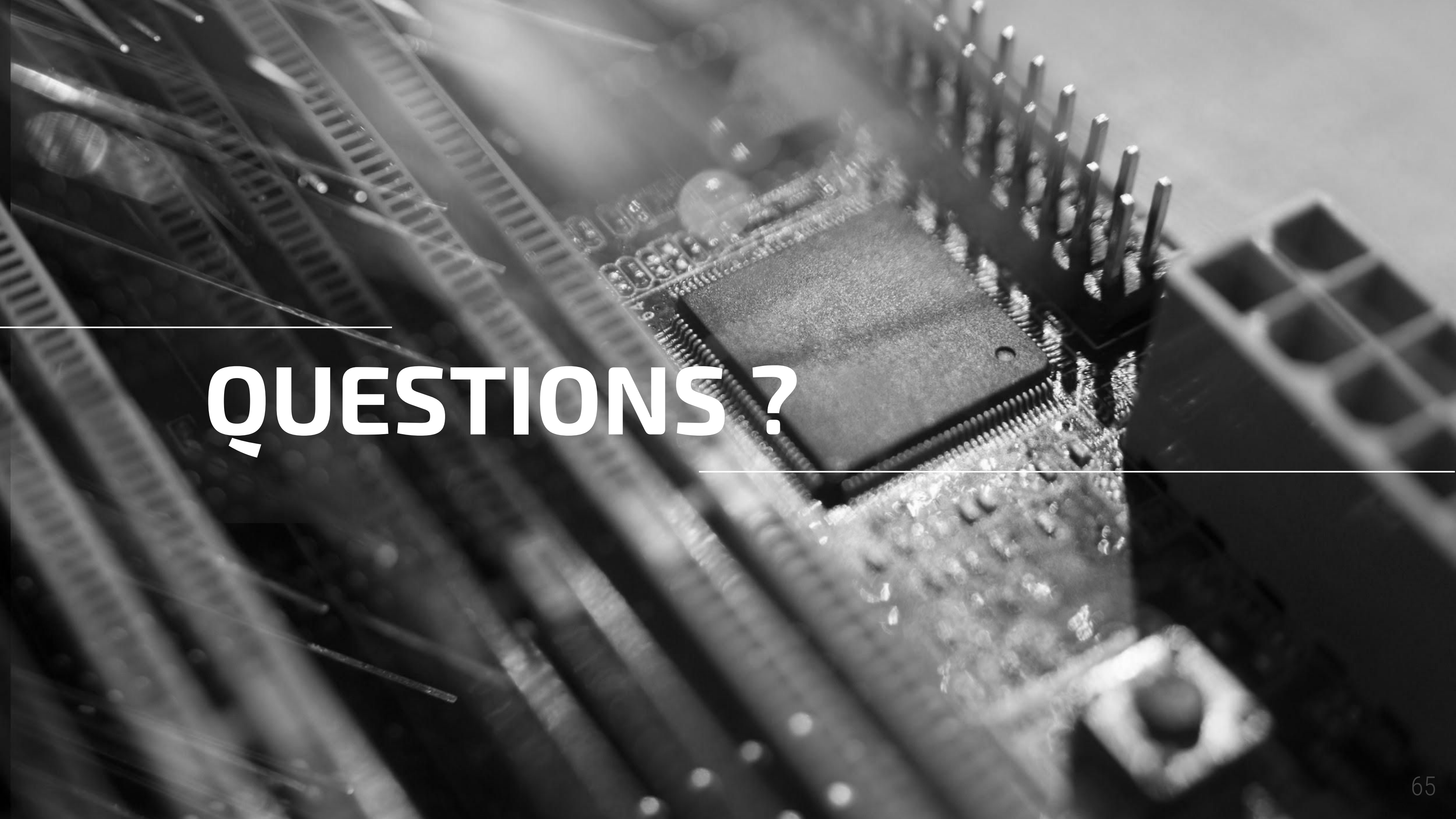
Conclusions (3)

- Hard to tell what is actually a false positive. (Anomaly does not equate to attack)
- Evolving normality.
- No domain specific knowledge, poor feature selection.

Conclusions (3)

- Hard to tell what is actually a false positive. (Anomaly does not equate to attack)
- Evolving normality.
- No domain specific knowledge, poor feature selection.

Plenty of room for improvements!



QUESTIONS ?