TREND ANALYSIS WITH ESA

DAVIDE VENEZIANO – ADVISORY SALES ENGINEER

RSA

Agenda

- Introducing the model
- How does it work?
- Implementation details

INTRODUCING THE MODEL

Analytics with the Netwitness Suite

- The Netwitness Suite already provides a number of ways to analyze your data:
  - Investigation
  - Reports
  - Big Data
  - Parallel Coordinates
  - Alerts
  - Correlation
  - Data Science
- **But can we do more?**

RSA

Trend analysis and baselining

- Why not having trend analysis capabilities as well?
  - Common request among our users
  - Implicitly available in ESA but without a formalized implementation
- When does it help?
  - Whenever a significant change in the rate of given value could imply a security issue
  - Not all the threats can be identified in this way!

RSA

Baselining model requirements

- Generic enough to work with any meta key
- Plug & play
- Reusable across multiple rules
- Suitable for production
- Minimum performance impact
- Persistent across service restart/reboot

RSA

Common challenges

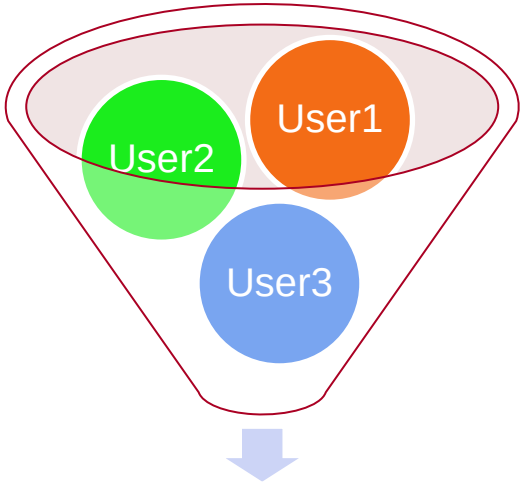
- To perform any statistical analysis, numbers are an obvious requirement
  - Have to be derived from the collected events first!
- A solid way to count the number of occurrences is key
  - Buffering in the ESA memory all the events with their meta for a long timeframe before applying any calculation would not scale
- A multi-stage approach is required to only store aggregated information in memory

RSA


HOW DOES IT WORK?

How does it work

- For each value of a given meta key, we will sum the number of occurrences:
  - This is done every minute
  - Then aggregated every 5 minutes
  - Then aggregated every hour
  - Then aggregated every day
- For each hour and for each value, a baseline is created:
  - Mean
  - Standard Deviation



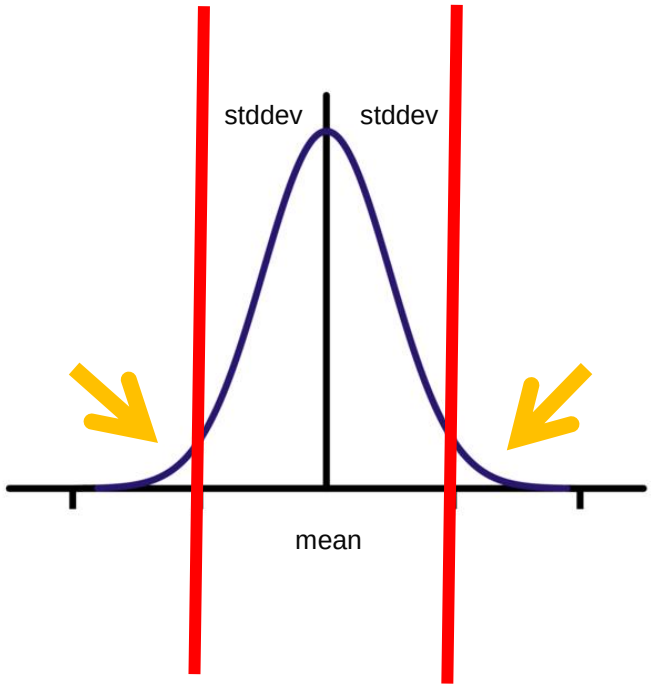
8am – 9am:

User1) mean=10; stddev = 2
User2) mean=23; stddev = 15
User3) mean=79; stddev = 5

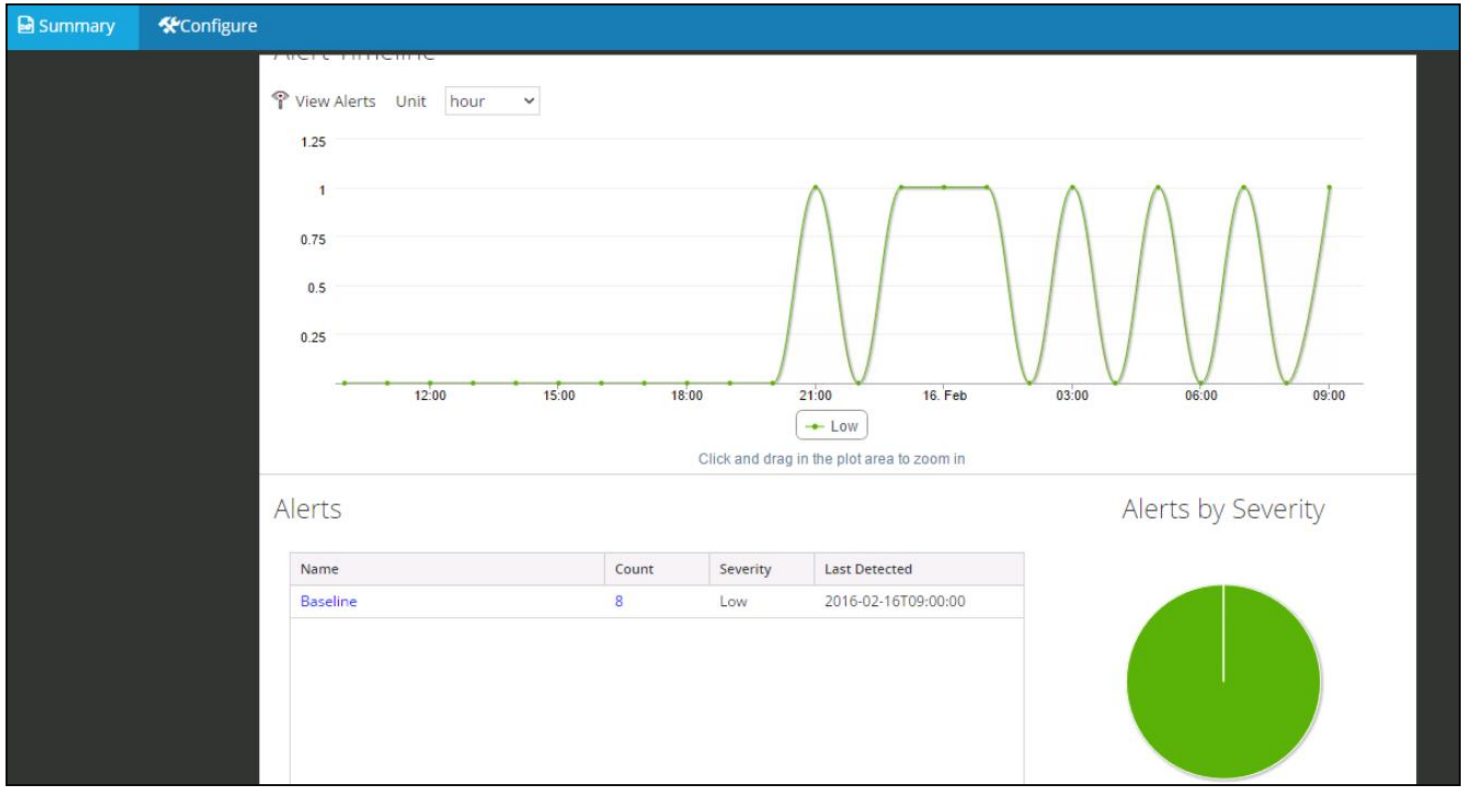
RSA

How does it work (cont.)

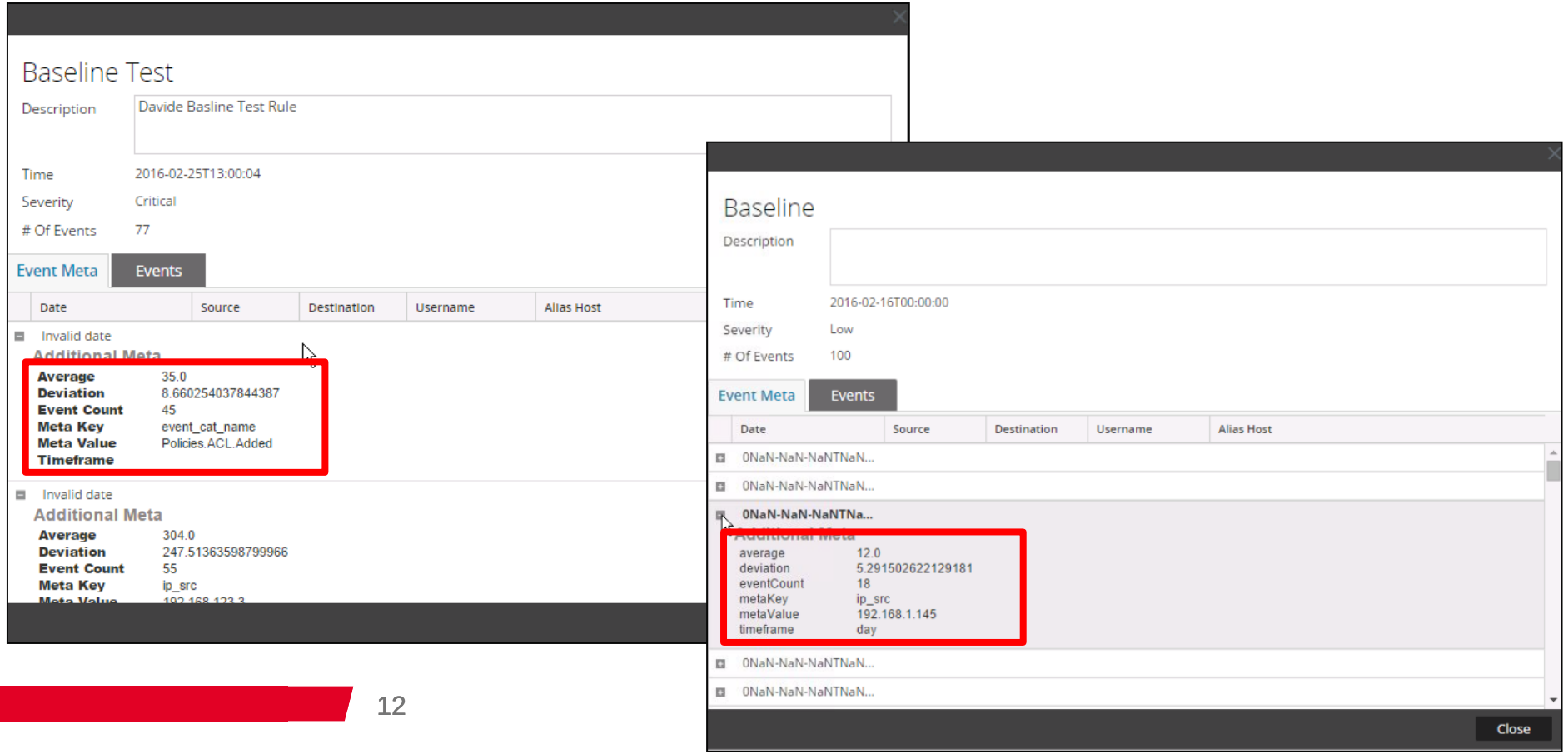
- Each hour
  - for each meta key
    - for each meta value
      - the number of occurrences is compared with the baseline of that hour
      - An alert is generated if there is a significant deviation
        - By default if the number of events is more than mean+sttdev or less then mean-sttdev
- The same is done for each day



Trend analysis module in action



Trend analysis module in action



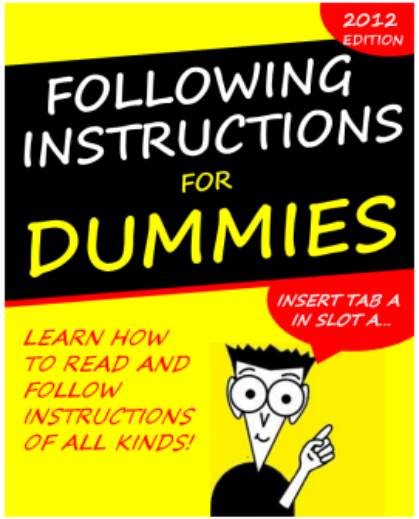
Possible Use Cases

- What does a significant change in the rate of given value mean?
  - Do NOT use meta keys with too many unique values (e.g. ip.src) since would generate too many false positives
  - Focus on those with a few but significant unique values:
    - Browsers - uncommon client may be associated with malicious codes
    - Country source/destination - can help identifying attacks or potential data exfiltration
    - TLDs - uncommon TLDs can be an indicator of something strange happening
  - Combine multiple meta keys together
    - E.g username and browser,

RSA

Implementing the model

1. Create a new ESA Advanced rule and copy and paste the EPL code provided in the last slide
2. Append or customize the «baselineInput» statement(s) to feed the model with the meta key you want to monitor
3. Activate trial mode and enable the rule
4. Constantly monitor in H&W the amount of memory the rule is consuming
5. Review the results at the end of the learning phase and increase/decrease «anomalyStddevTimes» as needed

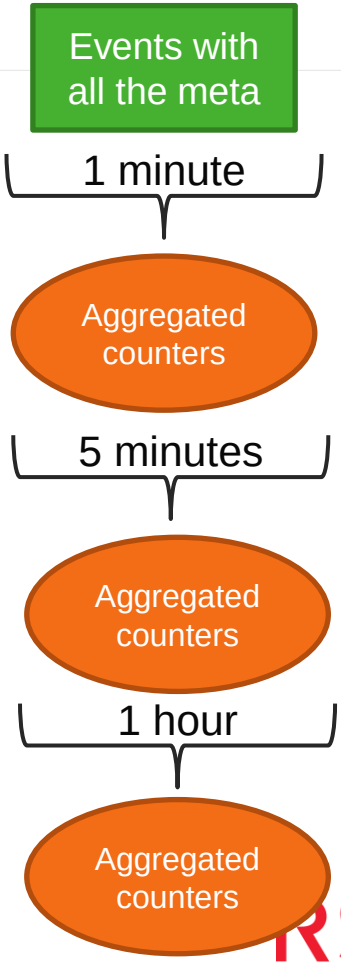


- The same approach can be leveraged in other custom rules whenever a very long timeframe has to be considered
- Combined keys can be monitored by concatenating them

RSA

Implementation overview

- Aggregation done in multiple steps:
  - Summarized data only is kept between each step to minimize performance impacts
- The baseline is created over the last 30 days:
  - Can be customized
  - Statistics are saved on disk to persist across reboot/restart
- Learning phase by default is 5 days:
  - Can be customized
  - This prevent alerting when no enough data is available
- Anomaly is by default mean +/- 3*sttdev
  - Can be customized



IMPLEMENTATION DETAILS

Trend analysis EPL implementation summary

1. Create time-based contexts
2. Create data structures to store aggregated statistics
3. Create a data structure to store the baseline statistics
4. Every minute the values of a given meta key count are fed them into the baseline model
5. Every 5 minutes / 1 hour / 1 day the data is aggregated, always for each meta value
6. A baseline model is generated at the end of each hour and day
7. Every hour, compare the number of events for each meta value with the baseline created for the same hour during the previous days and generate an alert if an anomaly is detected
8. Every day, compare the number of events for each meta value with the baseline created during the previous days and generate an alert if an anomaly is detected

Trend analysis EPL implementation

- Set the baseline model settings:

```
/* duration in days of the learning phase during which no alerts will be triggered */
CREATE VARIABLE integer learningPhaseInDays = 5;

/* the alert is triggered when the number of events is mean+/-anomalyStddevTimes*sttdev */
CREATE VARIABLE double anomalyStddevTimes = 3.0;
```


RSA

Trend analysis EPL implementation

- Create time-based contexts within which the calculations are performed:

```
CREATE CONTEXT contextEvery1Minute INITIATED @now AND PATTERN [every timer:at(*, *, *, *,
*)] TERMINATED AFTER 1 min;

CREATE CONTEXT contextEvery5Minutes INITIATED @now AND PATTERN [every timer:at(*/5, *, *,
*, *)] TERMINATED AFTER 5 min;

CREATE CONTEXT contextEvery1Hour INITIATED @now AND PATTERN [every timer:at(0, *, *, *,
*)] TERMINATED AFTER 1 hour;

CREATE CONTEXT contextEvery1Day INITIATED @now AND PATTERN [every timer:at(0, 0, *, *, *)]
TERMINATED AFTER 1 day;
```

Trend analysis EPL implementation

- Create data structures to store aggregated statistics lasting enough time to allow the sub-sequent data aggregation:

```
CREATE WINDOW baselineInput.win:time(5 minutes) (metaKey string, metaValue string,
eventCount long);

CREATE WINDOW baselineEvery5MinutesData.win:time(60 minutes) (metaKey string, metaValue
string, eventCount long);

@RSAPersist
CREATE WINDOW baselineEvery1HourData.win:time(30 days) (metaKey string, metaValue string,
eventCount long, timeframe string);

@RSAPersist
CREATE WINDOW baselineEvery1DayData.win:time(30 days) (metaKey string, metaValue string,
eventCount long, timeframe string);

@RSAPersist
CREATE WINDOW baseline.std:groupwin(metaKey, metaValue, timeframe).win:length(1)(metaKey
string, metaValue string, timeframe string, average double, dev double);
```


RSA

Trend analysis EPL implementation

- Every minute count the values of a given meta key and feed them into the model:

```
@Name('baselineInput1')
CONTEXT contextEvery1Minute

INSERT INTO baselineInput

SELECT 'event_cat_name' AS metaKey, event_cat_name AS metaValue, COUNT(*) AS eventCount

FROM Event(event_cat_name IS NOT NULL)

GROUP BY event_cat_name

OUTPUT snapshot WHEN TERMINATED;
```

Trend analysis EPL implementation

- Every 5 minutes the data is aggregated, always for each meta value:

```
@Name('baselineAggregateEvery5Minutes')

CONTEXT contextEvery5Minutes

INSERT INTO baselineEvery5MinutesData

SELECT metaKey, metaValue, SUM(eventCount) AS eventCount

FROM baselineInput

GROUP BY metaKey,metaValue

OUTPUT snapshot WHEN TERMINATED;
```


RSA

Trend analysis EPL implementation

- Every hour the data is aggregated again

```
@Name('baselineAggregateEvery1Hour')

CONTEXT contextEvery1Hour

INSERT INTO baselineEvery1HourData

SELECT metaKey, metaValue, SUM(eventCount) AS eventCount,
'hour_'||current_timestamp().get('hour').toString()  AS timeframe

FROM baselineEvery5MinutesData

GROUP BY metaKey,metaValue

OUTPUT snapshot WHEN TERMINATED;
```


RSA

Trend analysis EPL implementation

- At the end of the day, aggregate once again the data:

```
@Name('baselineAggregateEvery1Day') CONTEXT contextEvery1Day

INSERT INTO baselineEvery1DayData

SELECT metaKey, metaValue, SUM(eventCount) AS eventCount, 'day'  AS timeframe

FROM baselineEvery1HourData

GROUP BY metaKey,metaValue

OUTPUT snapshot WHEN TERMINATED;
```

Trend analysis EPL implementation

- At the end of the hour, calculate and store the baseline model for the previous hour:

```
@Name('baselineGenerateByHour')

INSERT INTO baseline

SELECT metaKey,metaValue,timeframe,average,stddev AS dev

FROM baselineEvery1HourData.std:groupwin(metaKey,metaValue,
timeframe).stat:uni(eventCount, metaKey,metaValue, timeframe)

GROUP BY metaKey,metaValue, timeframe

HAVING datapoints >= learningPhaseInDays;
```

Trend analysis EPL implementation

- Every hour, compare the number of events for each meta value with the baseline and generate an alert if an anomaly is detected:

```
@Name('baselineAlertByHour')
@RSAAlert(oneInSeconds=0)
CONTEXT contextEvery1Hour
SELECT baselineEvery1HourData.metaKey AS metaKey, baselineEvery1HourData.metaValue AS
metaValue, baselineEvery1HourData.eventCount AS eventCount,
baselineEvery1HourData.timeframe AS timeframe, baseline.average AS average, baseline.dev
AS deviation

FROM baselineEvery1HourData, baseline

WHERE baselineEvery1HourData.metaKey = baseline.metaKey AND
baselineEvery1HourData.metaValue = baseline.metaValue AND baselineEvery1HourData.timeframe
= baseline.timeframe AND (baselineEvery1HourData.eventCount >
(baseline.average+anomalyStddevTimes*baseline.dev) OR baselineEvery1HourData.eventCount <
(baseline.average-anomalyStddevTimes*baseline.dev)) AND
'hour_'||current_timestamp().get('hour').toString() = baselineEvery1HourData.timeframe
OUTPUT snapshot WHEN TERMINATED;
```


RSA

Trend analysis EPL implementation

- Every day compare the number of events for each meta value with the baseline and generate an alert if an anomaly is detected:

```
@Name('baselineAlertByDay')
@RSAAlert(oneInSeconds=0)
CONTEXT contextEvery1Day

SELECT baselineEvery1DayData.metaKey AS metaKey, baselineEvery1DayData.metaValue AS
metaValue, baselineEvery1DayData.eventCount AS eventCount, baselineEvery1DayData.timeframe
AS timeframe, baseline.average AS average, baseline.dev AS deviation

FROM baselineEvery1DayData, baseline

WHERE baselineEvery1DayData.metaKey = baseline.metaKey AND baselineEvery1DayData.metaValue
= baseline.metaValue AND baselineEvery1DayData.timeframe = baseline.timeframe AND
(baselineEvery1DayData.eventCount > (baseline.average+anomalyStddevTimes*baseline.dev) OR
baselineEvery1DayData.eventCount < (baseline.average-anomalyStddevTimes*baseline.dev))
OUTPUT snapshot WHEN TERMINATED;
```

Trend analysis EPL implementation

- Every day compare the number of events for each meta value with the baseline and generate an alert if an anomaly is detected:

```
@Name('baselineAlertByDay')
@RSAAlert(oneInSeconds=0)
CONTEXT contextEvery1Day

SELECT baselineEvery1DayData.metaKey AS metaKey, baselineEvery1DayData.metaValue AS
metaValue, baselineEvery1DayData.eventCount AS eventCount, baselineEvery1DayData.timeframe
AS timeframe, baseline.average AS average, baseline.dev AS deviation

FROM baselineEvery1DayData, baseline

WHERE baselineEvery1DayData.metaKey = baseline.metaKey AND baselineEvery1DayData.metaValue
= baseline.metaValue AND baselineEvery1DayData.timeframe = baseline.timeframe AND
(baselineEvery1DayData.eventCount > (baseline.average+anomalyStddevTimes*baseline.dev) OR
baselineEvery1DayData.eventCount < (baseline.average-anomalyStddevTimes*baseline.dev))
OUTPUT snapshot WHEN TERMINATED;
```

Full implementation

baseline_v2.1.txt

RSA

Thank You

RSA