Reconsidering Detection Engineering Approaches

Threat Specific Scoring for Prioritization

@praga_prag

- Reformed Red Teamer / Threat Hunter
- Blog:
  - Bouj33boy.com
- Former
  - DoD Red Team Operator
  - Adversary Tactics and Recon - Accenture
  - Endgame Hunt Training Instructor

- Hobbies
  - Astrophotography
  - Family Time

Prioritization Pain Points

- Lack of structure removes the organization's ability to prioritize
  - Analyst can choose any alert for any reason
- Structure of prioritization is based off assumptions
  - Priority of the alert is created from untested hypothesis
- Singular factor is used to determine priority
  - Priority of the alert is decided based off only one factor
- Structure of prioritization is never updated
  - A structure for scoring alerts exists but is never updated

Terms

- First, we need to define some terms specific to this talk
  - **Triage**: the concept of adding context to the base condition and establishing a priority according to the Funnel of Fidelity

  - **Subjective** metrics cannot be measured similarly by different people
    - Example: Providing a 1-5 scale on how good something is
  - **Objective** metrics are based on observable facts and should be the same regardless of the person
    - Example: Number of tickets closed by the SOC

Terms

- **A Priori**: Derived from Logic or Reason “To be known true”
  - Example: “Roosters crow when the sun rises.”
- **A Posteriori**: Derived from observed facts “To be observed to be true”
  - Example: “The sun rises after the rooster crows.”

- **Both** methods are utilized in prioritizing, however both methods can create misconceptions

Base Condition

- To effectively prioritize malicious events we must first derive the base condition.
  - Example: To identify malicious service creations, we need telemetry for **all** service creations.
    - The collection method must be first considered.
      - Example: Service Creation = **Event ID 4697** vs **Raw Registry Event Data**

Base Condition

Base Context

Registry key creation under the ***HKLM\SYSTEM\CurrentControlSet\Services*** key

Service Creation (Detection)

Author: Jonathan Johnson // Jared Atkinson

Goal:

To alert when a new value is made within the `Services` registry key.

Base Condition:

Registry key creation under the `HKLM\SYSTEM\CurrentControlSet\Services` key

Analytic:

```
import splunklib.results as results
#Query:
query = "search index=main sourcetype=WinEventLog:Microsoft-Windows-Sysmon/Operational EventCode=12 earliest=-230m TargetObject=HKLM\\\\System\\\\CurrentControlSet\\\\Services\\\\*"
query_results = service.jobs.oneshot(query, count=0)
reader = results.ResultsReader(query_results)

results = []

for result in reader:
    results.append(result)

df_EID_12=pd.DataFrame(results)
```

Base Context:

- Service Name Created
- Image that created the service
- ProcessGUID of the creating process
- Timestamp

Base Condition

- Service Creation Base Condition:
    - Service Name Created
    - Image that Created the Registry Key
    - ProcessGUID of the Creating Service
    - TimeStamp

- We don’t have enough data to make a triage decision
    - We need to add factors to make our decision

Factors

- We must add contextual evidence “factors” to the base condition to enable decision making capability
  - Consideration #1:
    - Our ability to use contextual factors in making decisions in predicated by which factors to include
  - Consideration #2:
    - We can’t determine if the factor aids in decision making if we don’t include the factor

Identifying Factors: A Priori

- In order to identify factors to add to the base condition we must make logical assumptions about malicious behavior
  - Example: Service Creation
    - Assumption #1: Remote service creations are a malicious factor
    - Assumption #2: Binary creating the registry key should be services.exe
    - Assumption #3: Autostart services are a malicious factor
    - Assumption #4: Processes requesting to create services should be sc.exe

Identifying Factors: A Priori

- Assumption #1: Remote service creations are a malicious factor
    - Hypothesis is derived from knowledge of the Post Exploitation phase of the kill chain

- **Remote** service creation accomplishes 3 goals:
    - Lateral Movement
    - Persistence
    - Execution

- **Local** service creation accomplishes 2 goals:
    - Persistence
    - Execution

- We can assume that an adversary would rather accomplish more goals with remote service creation than less goals with local service creation

Identifying Factors: A Priori

- Assumption #2: Binary creating the registry key should be services.exe
    - Hypothesis is derived from research into the underlying technology
  - To create a service, specific criteria that are normally met:
    - A registry key must be created within *HKLM\SYSTEM\CurrentControlSet\Services*
    - 5 registry values are set within *HKLM\SYSTEM\CurrentControlSet\Services*
    - These changes to the registry are normally made by ***services.exe*** at the request of another process

Identifying Factors: A Priori

- Assumption #3: Autostart services are a malicious factor
  - Hypothesis is derived from the assumption that adversaries want to successfully survive a reboot ***autonomously***
    - **An autostart services** enables the adversary to execute code on system startup
    - **A manual start-type service** would provide little value to an adversary as the service would require interaction for code execution

Identifying Factors: A Priori

- Assumption #4: Processes requesting to create services should be sc.exe
    - Hypothesis is derived from default characteristic of using ***sc.exe*** to request that services.exe create a service
        - *This assumption represents most organizations' allowlist based malicious factor*
        - This assumption is using the deviation of the baseline of Windows native binaries and their default behavior

Proving Factors: A Posteriori

- **Validation and testing** are required to take these assumptions (A priori) and turn them into **empirical evidence** (A posteriori)
  - Empirical means “by observation” or through validation and testing.
    - Example:
      - You can say the ocean is made of water, because you have observed both water and the ocean.

Proving Factors: A Posteriori

- To validate our hypothesis, we need to collect our data set and apply data analytic techniques to identify the differences in the criteria that consistently differentiate legitimate from malicious
- We will need to account for the portions of our data set that may skew our results:
    - Test Pool Size
    - Single or Multiple Environments
    - Variations in Multiple Environments

Proving Factors: A Posteriori

- Descriptive Statistics
  - Measures of Frequency
  - Measures of Tendency
  - Measures of Variation
  - Measures of Position
- Inferential Statistics
  - Bayes' Theorem
  - Analysis of Variance
  - Regression

Proving Factors: A Posteriori

- Measures of Frequency:
  - Example:
    - **Assumption #1**: Remote service creations are a malicious factor
      - The prevalence of locally created services across three environments outweighs remotely created services 1000/1.
    - **Conclusion**: Legitimate remote service creation is very rare across three environments
      - We will score the occurrence of Sysmon Event ID 12 - registry key creation "HKLM\SYSTEM\CurrentControlSet\Services" with a correlated Sysmon Event ID 3 - Network Connection Initiated with a score **higher** than that of a locally created service.

Proving Factors: A Posteriori

- **Measure of Frequency** feeds into **Probability**
  - **Assumption #2:** Binary creating the registry key should be services.exe

Experiment

Binary creating the registry should be services.exe

→

Outcome

Two binaries used in multiple occurrences of the services registry key creation

→

Event

Event 1: The binary was services.exe

Event 2: The binary was reg.exe

→

Probability

95% of occurrences were Event 1 = +.5 Score

5% of occurrences were Event 2 = 9.5

Proving Factors: A Posteriori

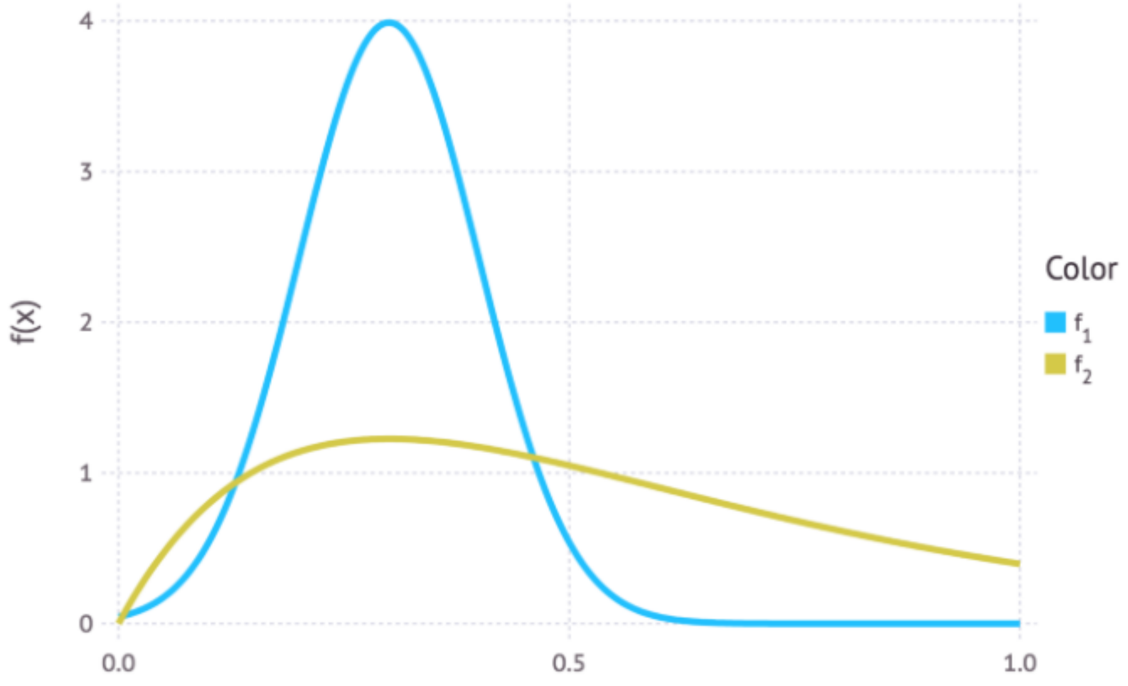
- Bayes' Theorem
  - **Assumption #3**: Autostart services are a malicious factor
  - Let's ask "What is the probability that the malicious service will be a Autostart type?"
    - P(A = Number of autostart services as compared to ALL service creation)
    - P(B = Triage Result)
    - P(A = number of autostart services = 13/52 Services Created were Autostart = 25%
    - Based on this historical data we can say there is a 25% chance that a service will be autostart without knowing if it will be malicious or not.

Proving Factors: A Posteriori

- Statistical Inference
  - Deducing probability from data using Bayes' Theorem
  - Relies on "marginal probability"
  - For the last example of historical data in our data set, the percentage of autostart type services was 25%. However, in another data set we see the that there have been as many as 40%.
    - We now know the autostart services can be as many as 40% and as low as 25%.

Proving Factors: A Posteriori

- Autostart services can be as many as 40% and as low as 25%.
- **We can infer that the likelihood for our autostart service creation event to be useful as malicious criteria because it is a deviation from baseline as HIGH AS 75% or as low as 60%**

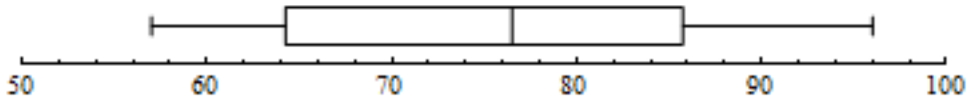


Proving Factors: A Posteriori

- Measures of Position
    - This approach is ideal for multiple data sets
  - Determine the scores for the **five-number summary**
    - Provides information on the average, the dispersion, and the shape
    - The five numbers are the minimum, the first quartile, the median, the third quartile, and the maximum

Proving Factors: A Posteriori

- Measure of Position:
  - Assumption #4: The processes requesting to create services should be sc.exe
  - Example: Based on the scores the minimum, the first quartile, the median, the third quartile, and the maximum is: 57, 64.25, 76.5, 85.75, and 96



Proving Factors: A Posteriori

- Once we have identified the differences in criteria, we must then identify the behavior relationships to begin to form our empirical scoring
  - The empirical score is derived from the evidence of the statistical analysis
  - The empirical scoring must be documented so that future analysts can research, tune and account for variants in the score.

Continuous Improvement

- How do we prevent stagnating on our conclusions?
  - Empirical scoring requires constant testing to validate hypothesis and create conclusions
    - Periodic validation of hypothesis and the correlated empirical evidence must be accomplished
      - Ideally, this is where some **machine learning** could be utilized to validate that metrics and conclusions remain the same over time

Continuous Improvement

- There is to be expected some margin of error
  - Bias agnostic testing procedures must be validated to avoid skewed or stagnate results
    - All analysts have some bias – we must account for that with a documented margin of error
    - We also need to account for our incomplete knowledge
      - Of all the services we have; what do they have in common and is that local to our environment or globally all environments

References and Resources

- Jared Atkinson
  - Research: https://posts.specterops.io/@jaredcatkinson
  - Funnel of Fidelity: https://posts.specterops.io/introducing-the-funnel-of-fidelity-b1bb59b04036?gi=9b5b23dbd69b
- Jonny Johnson
  - Projects: https://github.com/jsecurity101/
  - Research: https://posts.specterops.io/@jsecurity101
- Josh Prager
  - Research: https://posts.specterops.io/@bouj33boy

- Statistical Analytics
  - https://sciencing.com/similarities-of-univariate-multivariate-statistical-analysis-12549543.html
  - https://towardsdatascience.com/probability-concepts-explained-bayesian-inference-for-parameter-estimation-90e8930e5348
  - http://www.milefoot.com/math/stat/desc-positions.htm
  - https://www.analyticsvidhya.com/blog/2017/02/basic-probability-data-science-with-examples/
  - https://baselinesupport.campuslabs.com/hc/en-us/articles/204305685-Inferential-Statistics

www.specterops.io
@specterops
info@specterops.io