ProcAID: Process Anomaly-based Intrusion Detection

AJ Read

ShmooCon
March 24th, 2022

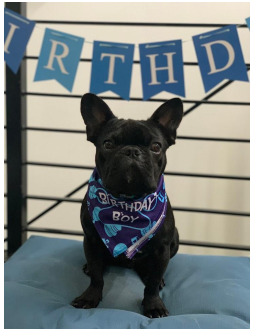
THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Introduction

- Graduate Student at GW
  - MS in Cybersecurity in Computer Science
  - GraphLab: DARPA CHASE Project
- Active Duty Coast Guard Officer

GW SCHOOL OF ENGINEERING & APPLIED SCIENCE

DARPA DEFENSE ADVANCED RESEARCH PROJECTS AGENCY



Agenda

- Motivation and Challenges
- Methodology
- Results
- Future Work
- Implications and Conclusion
- Questions

Agenda

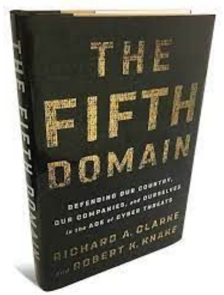
- Motivation and Challenges
- Methodology
- Results
- Future Work
- Implications and Conclusion
- Questions

Motivation

- Advanced Persistent Threat (APT) success in current landscape
    - Nation-state actors
    - Well-funded and well-staffed
- Offensive preference in infosec

Challenges

- First intrusion detection model, invented by Dorothy Denning
    - Definition: any deviation in normal operations on a system
- State of intrusion detection
    - Signature-based vs. Anomaly-based
- APTs have breached the capabilities of current assets



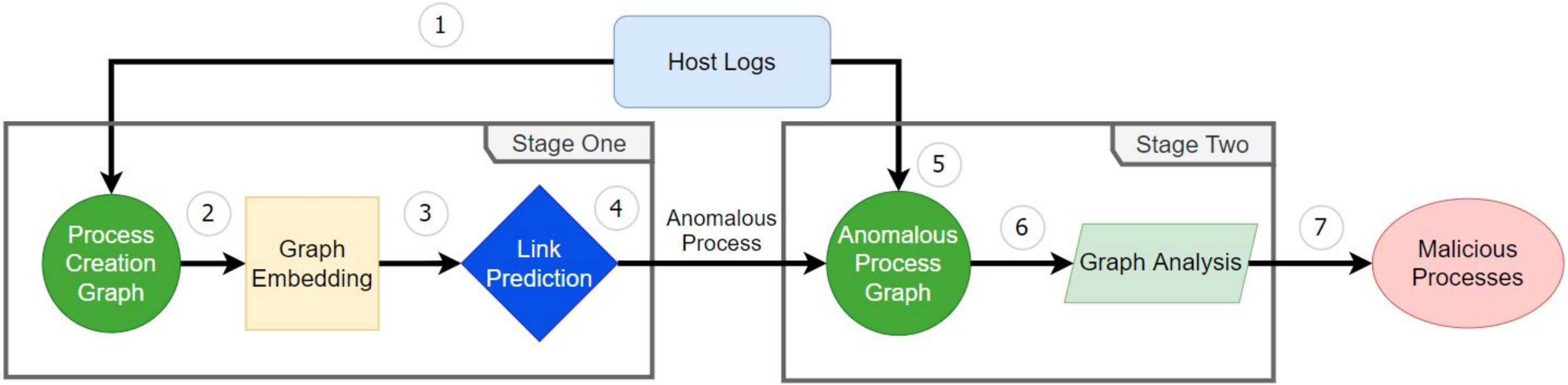
Agenda

- Motivation and Challenges
- Methodology
- Results
- Future Work
- Implications and Conclusion
- Questions

ProcAID Methodology

<u>Stage One:</u> Unsupervised Anomaly Detection via Link Prediction

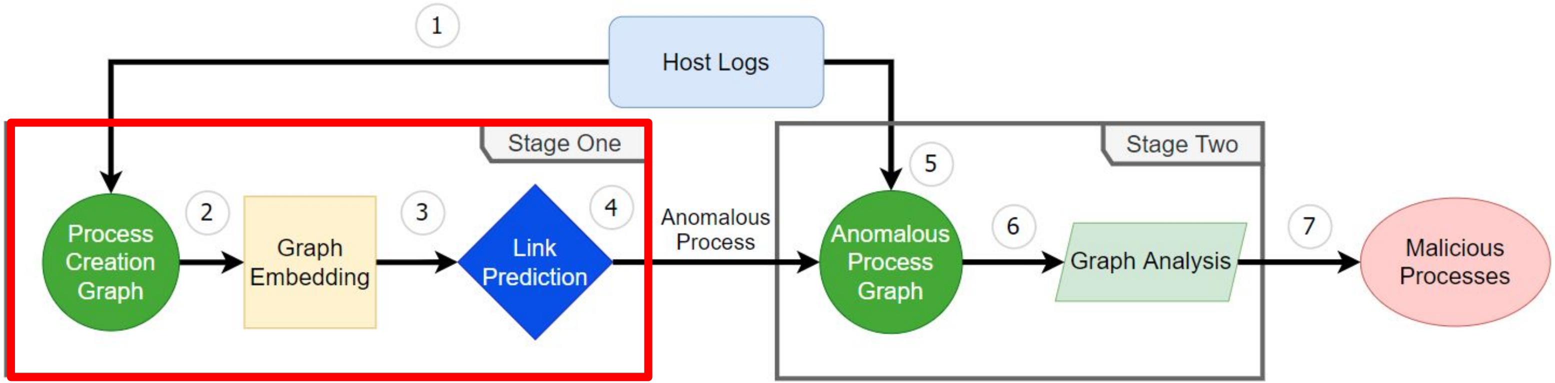
<u>Stage Two:</u> Inverse Graph Leadership and Inverse Graph Density Analysis



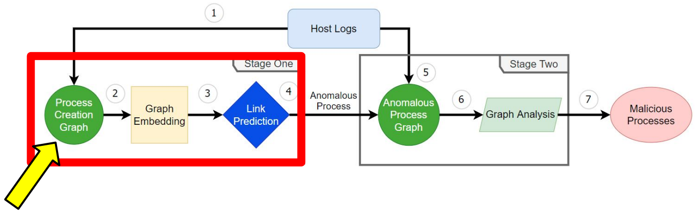
ProcAID Stage One

- **Goal:** Find anomalous process creations
- Key Characteristics:
  - Node2Vec Embedding
  - Logistic Regression

- **Method:** Model user and process interactions
  - Process tree analysis and evaluation
  - Creation of relationships between features

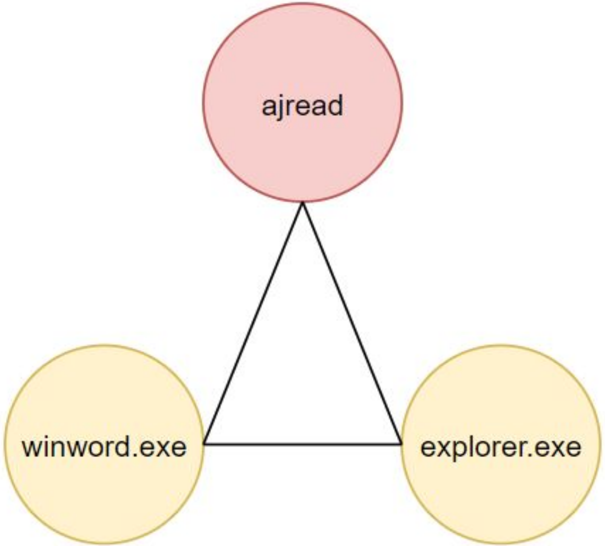


Stage One: Graph Creation

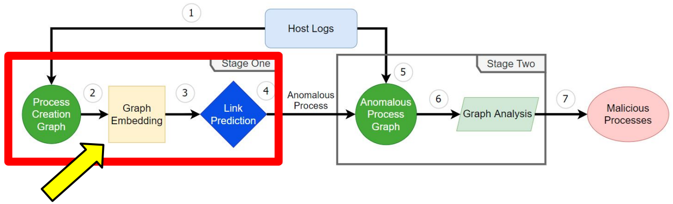


- Form two Process Creation Graphs (train and test) with host logs
- Graph Schema
  - Nodes: User, Process Path, or Parent Process Path
  - Edges: Executions/interactions

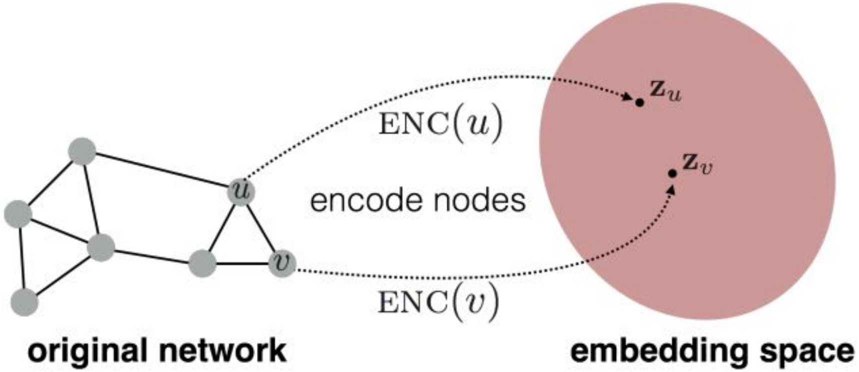
Example: User "ajread" spawns "winword.exe" from "explorer.exe"



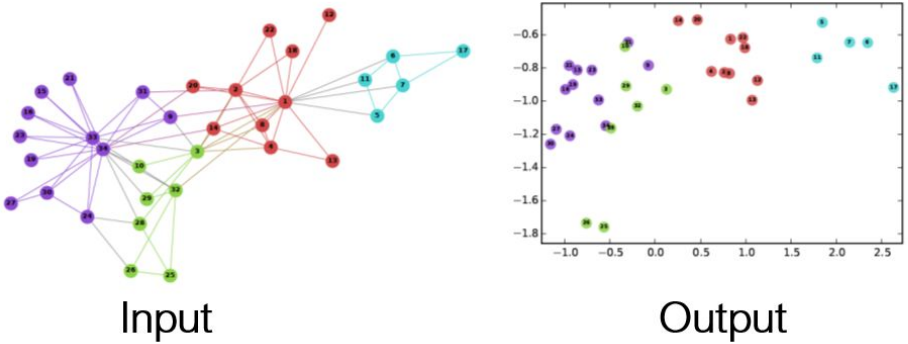
Stage One: Graph Embedding



- Learn information from Process Creation Graphs through embedding
- *node2vec: Scalable feature learning for networks*[1]
  - Second order bias parameters
  - Hadamard embedding of edges



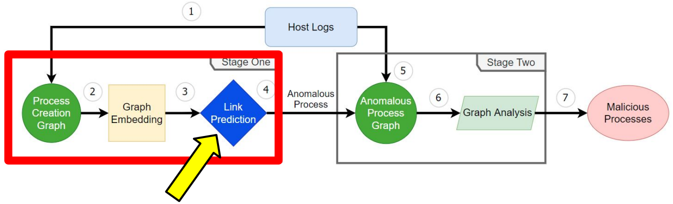
http://snap.stanford.edu/proj/embeddings-www/files/nrltutorial-part1-embeddings.pdf



http://snap.stanford.edu/proj/embeddings-www/files/nrltutorial-part1-embeddings.pdf

[1]A. Grover and J. Leskovec, “node2vec: Scalable feature learning for networks,” in Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 855–864, 2016.

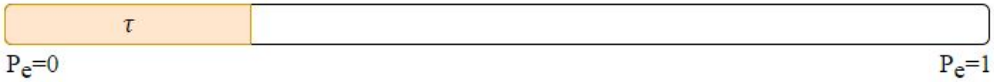
Stage One: Link Prediction



- Predict existence of test graph edges
- ML Algorithm: Logistic Regression
  - Quick training time
  - Well calibrated probabilities
    - *Pe*=0, edge does not exist
    - *Pe*=1, edge does exist

$$p_e = \frac{1}{1 + \exp^{-x*T}}$$

- Prediction Threshold (τ)
  - Edge anomaly if probability less than threshold



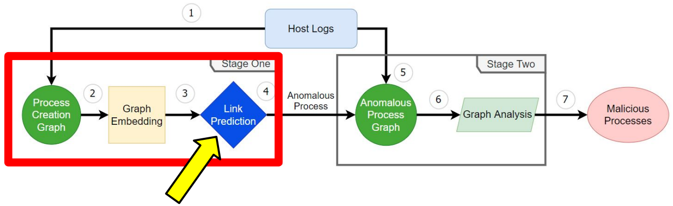
$$e_{anom} = \begin{cases} 1 & p_e < \tau \\ 0 & \text{otherwise} \end{cases}$$

Dataset

- Operationally Transparent Cyber Data Release (OpTC)
- 1000 hosts with multiple days of benign and malicious activity
- Logs formulated into “object” and “action” pairs for analytics

Host	Type of Exploitation	Post-Exploitation Actions
0201	Batch file containing Powershell code	PowerShell Empire, Mimikatz, registry edits
0501	Phishing with Macro-enabled Word Document	DeathStar, PowerShell Empire, Windows Management Instrumentation (WMI) subscriptions, SSH forwarding

Stage One Evaluation Results



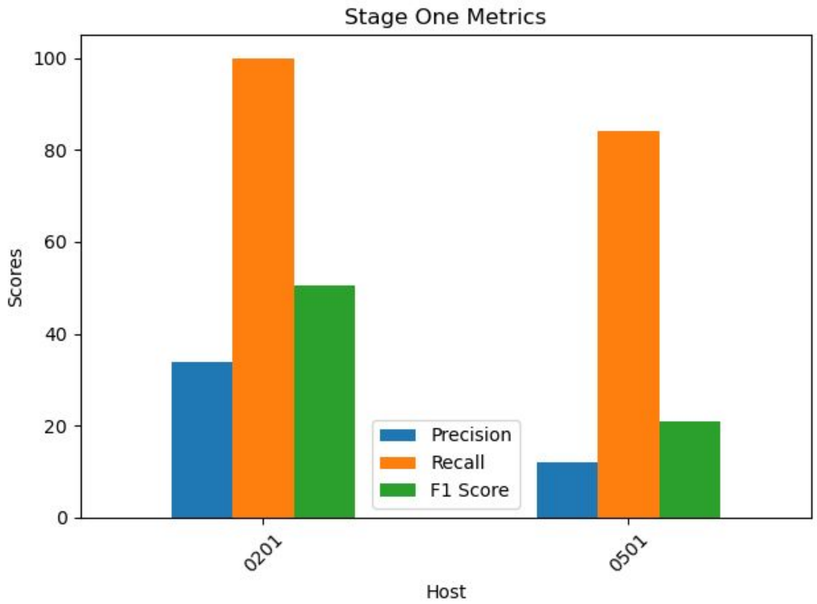
Host 0201

- ProcAID discovers all malicious process creation events
- False positives impact precision

Host 0501

- ProcAID discovers majority of malicious process creation events
- Overwhelmed by false positives

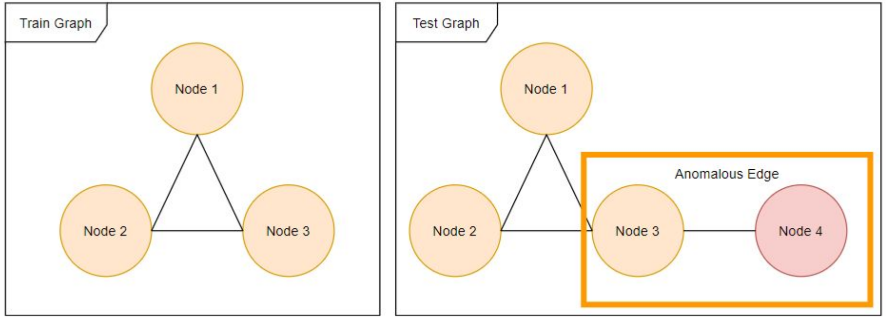
<u>**Conclusion:**</u> Engineer Stage Two to intelligently filter false positives



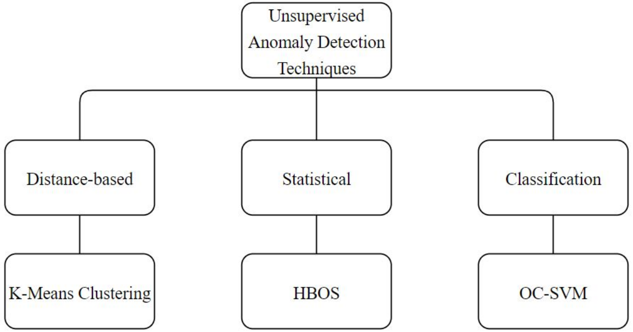
Host	Precision	Recall	F1 Score
0201	33.871	100.00	50.602
0501	11.852	84.211	20.779

Algorithm Comparison

- Independent Variables:
  - Training and Testing Data
  - Features: user, process path, and parent process path
- NewEdge Graph Algorithm
  - Returns new edges found in the Test Graph that are **not** in the Training Graph
    - Key: No threshold

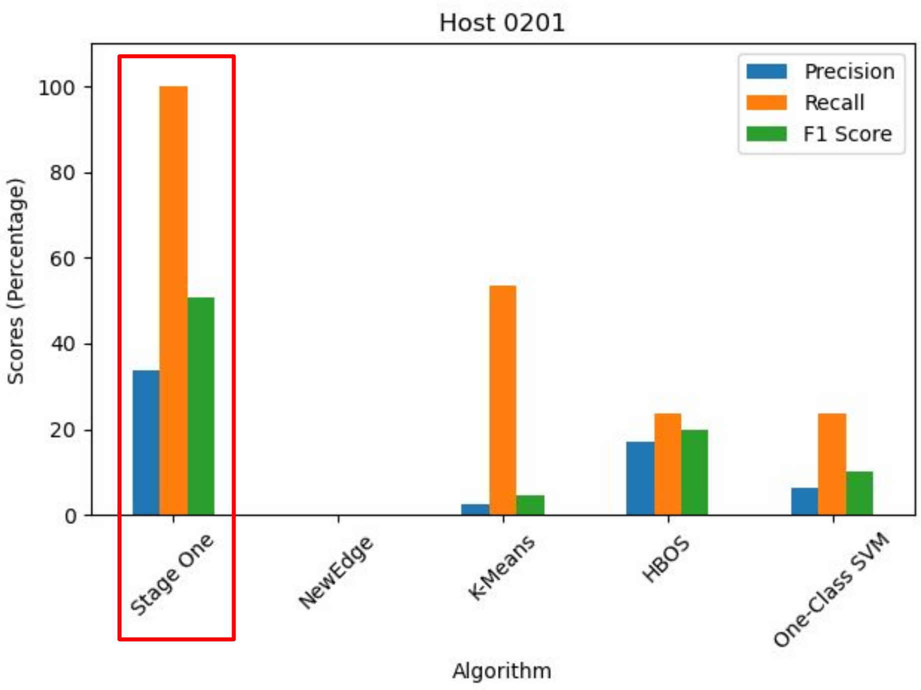


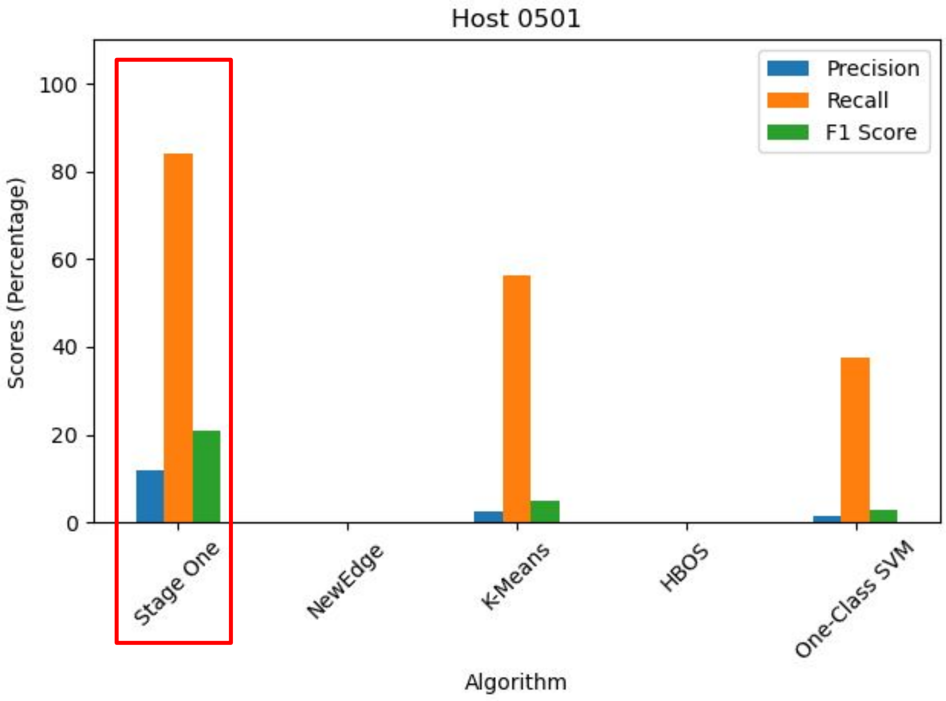
- Unsupervised Algorithms[2]:
  - Distance-based: K-Means Clustering
  - Statistical: Hierarchical Based Outlier Score (HBOS)
  - Classification: One-Class Support Vector Machine (OC-SVM)



[2] F. Falcão, T. Zoppi, C. B. V. Silva, A. Santos, B. Fonseca, A. Ceccarelli, and A. Bondavalli, “Quantitative Comparison of Unsupervised Anomaly Detection Algorithms for Intrusion Detection,” in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pp. 318–327, 2019.

Algorithm Comparison Results (Stage One)

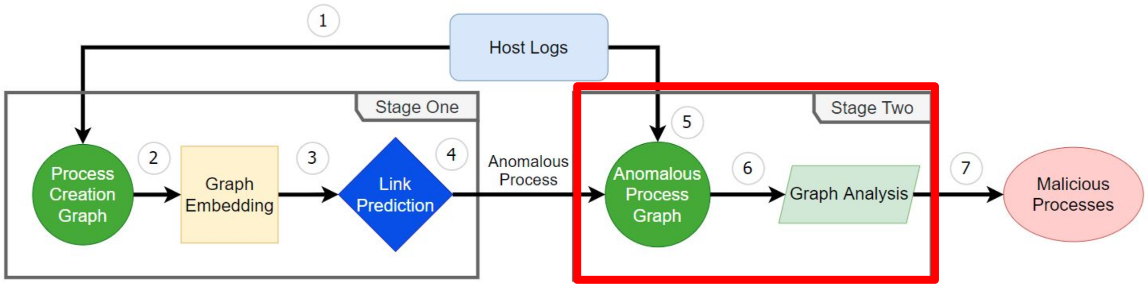




ProcAID: Stage Two

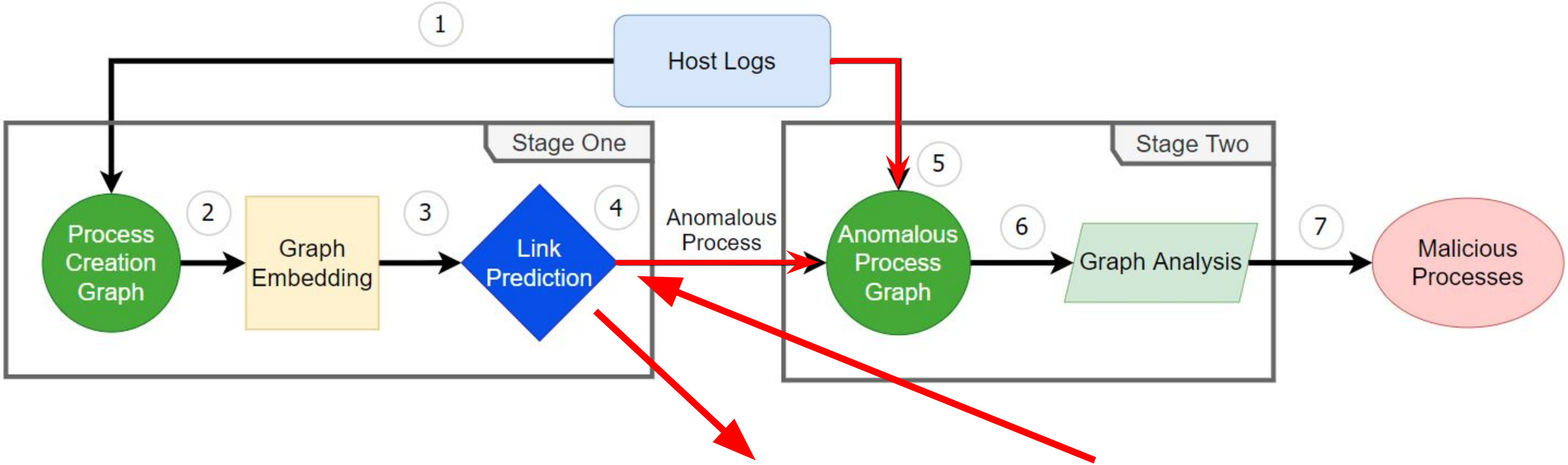
- **<u>Goal:</u>** Scrutinize anomalous process from Stage One

- **<u>Method:</u>** Examine anomalous edges using graph analytics



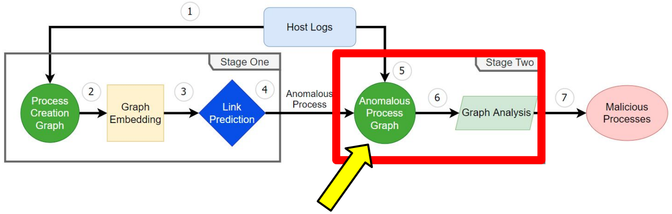
- Substages
  - Data Preparation
    - Format data for Graph Creation
  - Graph Creation
    - Anomalous Process Graph creation
  - Analysis
    - Inverse Graph Leadership
    - Inverse Graph Density

Stage Two: Example



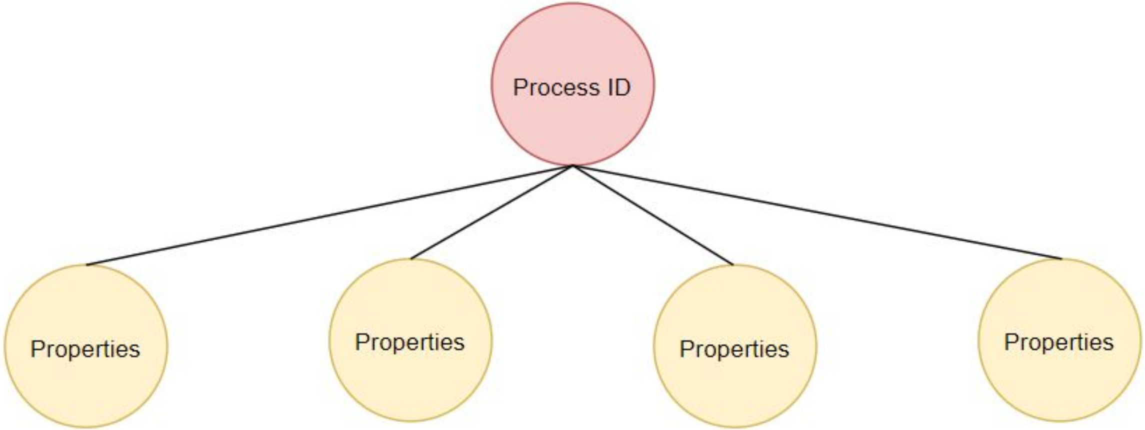
ajread—winword.exe → PID: 926

Stage Two: Graph Creation



- **<u>Purpose:</u>** Model process entire interaction with host

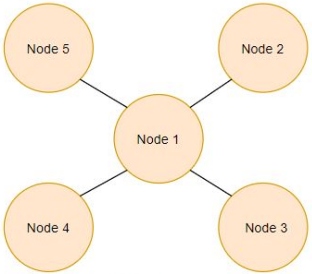
- Graph Schema
  - Nodes: PID, Parent PID, Registry Values, Registry Keys, Source IP, Destination IP
  - Edges: Interactions



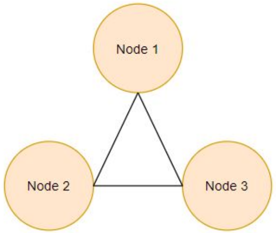
Analysis Background

- Leadership:
  - Measure of the extent of which a graph is dominated by a single node

$$L = \frac{\sum_{i=1}^{n} d_{max} - d_i}{(n-1)(n-2)}$$



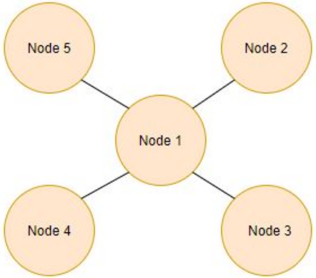
Maximal



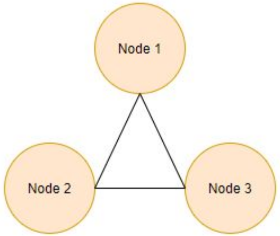
Minimal

- Density
  - Measure of the number of connections between nodes in comparison to the number of possible connections between nodes

$$D = \frac{2m}{n(n-1)}$$

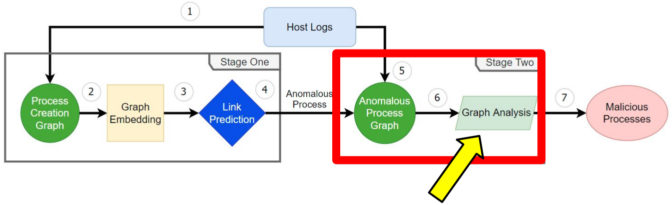


Density=0.40



Density=1.0

Stage Two: Analysis



Assumption 1:

- The Anomalous Process Graph for a malicious process will have a high inverse graph leadership value because process execution will be dominated by multiple objects.

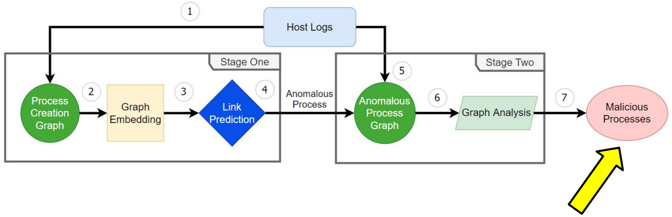
$$L^{-1} = \frac{(n-1)(n-2)}{\sum_{i=1}^{n} d_{max} - d_i}$$

Assumption 2:

- The Anomalous Process Graph for a malicious process will have a high inverse graph density value because objects will interact with a wide range of unique subjects during execution.

$$D^{-1} = \frac{n(n-1)}{2m}$$

Final Maliciousness Score



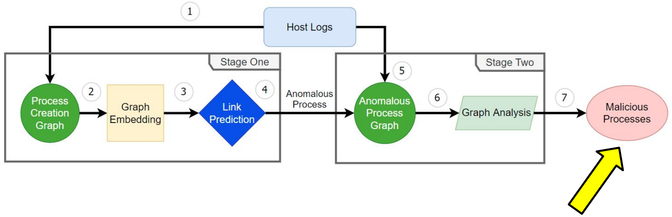
Assumption 3:

- The total maliciousness score for a malicious process will be higher than the total maliciousness score for a benign process.

$$MalScore_{process} = \sum_{i=0}^{N} [L^{-1}[i] + D^{-1}[i]]$$

Agenda

- Motivation and Challenges
- Methodology
- Results
- Future Work
- Implications and Conclusion
- Questions



Results for Host 0201

- Placement of malicious activity at the highest percentiles of the results
  - Assumption 3 is affirmed
- Effectively filters false positives from Stage One
- Average Run Time: 30.966 sec
  - Stage One : ~5 sec

Threshold	Precision	Recall
Top 1	1.000	0.500
Top 5	0.800	0.800
Top 10	0.600	0.857
Top 15	0.467	1.000
Top 20	0.400	1.000

Table 4.8: Top-K Comparison for Host 0201

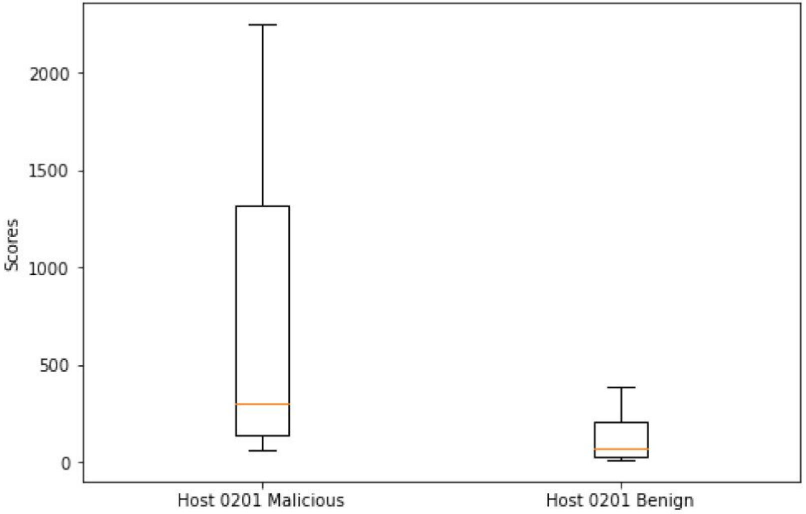
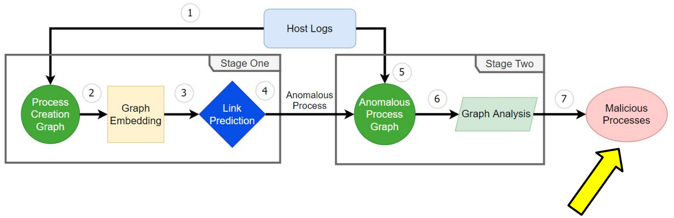


Figure 4.9: Box and Whisker Plot for Scores on Host 0201



Results for Host 0501

- Increased number of false positives impact results
- Placement of malicious activity at higher percentiles of results
  - Assumption 3 is affirmed
- Average Run Time: 268.23 sec
  - Stage One: ~5 sec

Threshold	Precision	Recall
Top 1	0.000	0.000
Top 5	0.800	1.000
Top 10	0.400	1.000
Top 15	0.267	1.000
Top 20	0.200	1.000

Table 4.10: Top-K Comparison for Host 0501

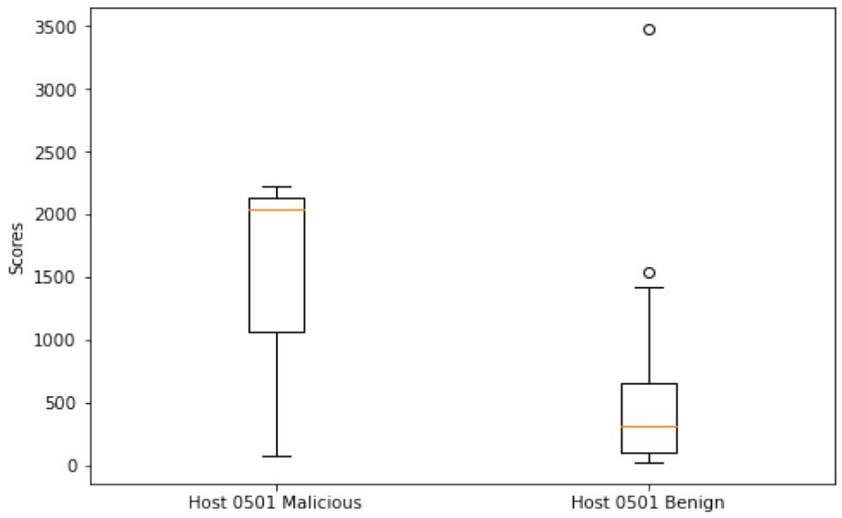
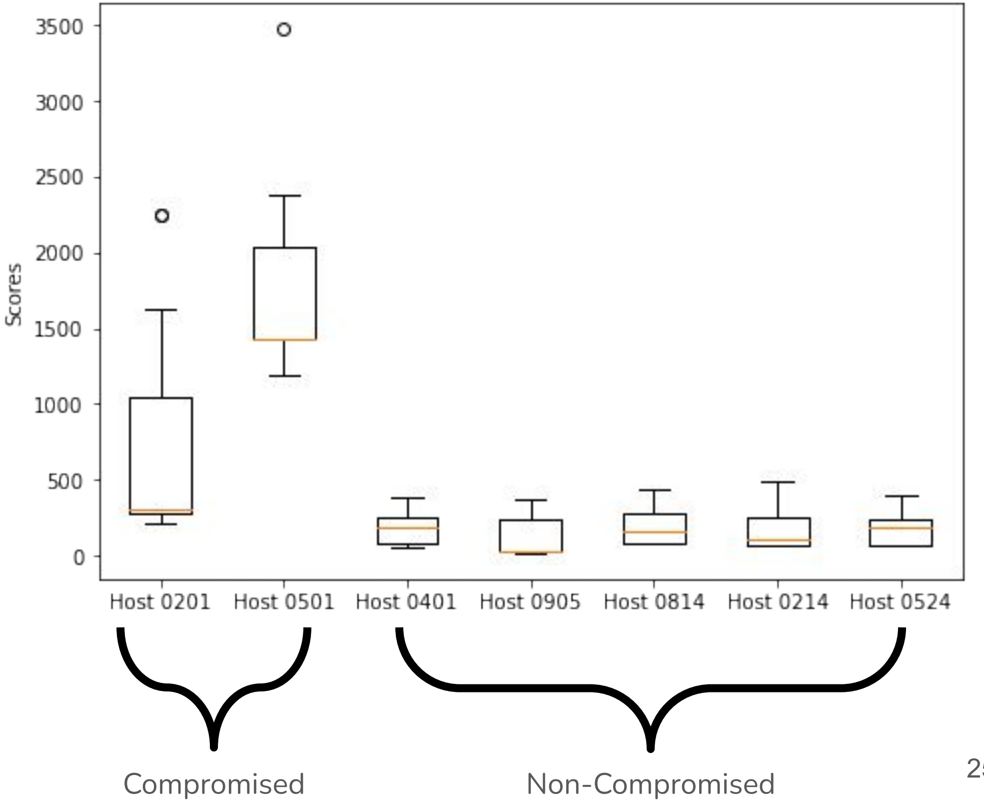


Figure 4.10: Box and Whisker Plot for Scores on Host 0501

ProcAID Results Across Multiple Hosts

- Enterprise Implementation
- Scores reflect both malicious and benign processes
- Compromised hosts show clear increased mean and standard deviation



Agenda

- Motivation and Challenges
- Methodology
- Results
- Future Work
- Implications and Conclusion
- Questions

Future Work

- Training and testing time analysis
- Graph embedding techniques other than Node2Vec
    - Other techniques could address the offline/online issue
- Datasets
    - Any verbose dataset with Windows Security Event ID 4688 or similar
    - Real world vs. Academic Datasets
- Full enterprise implementation
    - Placement of users and administrators based on process creation activity in Stage One

Agenda

- Motivation and Challenges
- Methodology
- Results
- Future Work
- Implications and Conclusion
- Questions

Implications and Conclusions

Implications

- ProcAID application is simple and vast in the cybersecurity space
  - No rule creation, no supervision
  - Offline solution
  - Training required

Conclusion

- Fusion of unsupervised link prediction, inverse graph leadership, and inverse graph density
- Efficient and effective host-based anomaly detection system for combatting APTs
- Applications of graph theory in information security

Publication

- https://www.proquest.com/dissertations-theses/procaid-process-anomaly-based-intrusion-detection/docview/2604788938/se-2

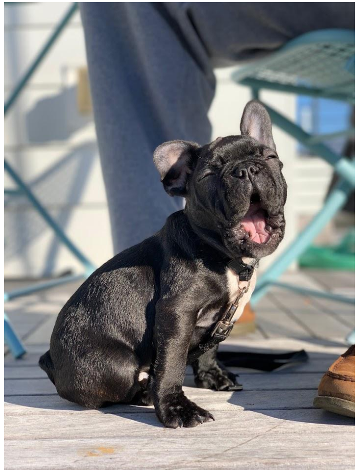
Social Media

read.austin@gmail.com

ajread4@github.com

twitter.com/ajread3

Questions?



Supplemental Slides

ProcAID Algorithm

- Ingests all host logs
- Returns anomalies with maliciousness scores
- Implemented in Python
  - NetworkX
  - SKLearn
- Stage One
  - Lines 1-4
- Stage Two
  - Lines 6-20

Algorithm 1 ProcAID Algorithm

```
Input: HostLogs
Output: Anom,MalScore
 1: //Stage One
 2: ProcGraphTrain=CreateGraph(CreateProcTrain)
 3: ProcGraphTest=CreateGraph(CreateProcTest)
 4: ProcAnomalies=GraphAnomaly(ProcGraphTrain,ProcGraphTest,Thres)
 5: //Stage Two
 6: MalEdgeCollection=FindLogs(ProcAnomalies)
 7: for edge in MalEdgeCollection do
 8:   for parentproc in edge do
 9:     ParentProcData=SplitTime(parentproc,Time)
10:     EvaluationGraph=CreateAnomalyGraph(ParentProcData)
11:     EdgeDataParentProc=LeadershipDensity(EvaluationGraph)
12:   end for
13:   for proc in edge do
14:     ProcData=SplitTime(proc,Time)
15:     EvaluationGraph=CreateAnomalyGraph(ProcData)
16:     EdgeDataProc=LeadershipDensity(EvaluationGraph)
17:   end for
18:   Anom, MalScore=CombineData(EdgeDataParentProc,EdgeDataProc)
19: end for
20: return Anom, MalScore
```

ProcAID Specifications

1. Node2Vec
   a. Number of walks: 100
   b. Walk length: 5
   c. P: 0.125
   d. Q: 2.0
2. Stage One Output:
   a. Host 0201: 15 Edges, Host 0501: 48 Edges
3. Graph Sizes
   a. Process Creation (Train/Test): ~100 Nodes and ~300 Edges
   b. Anomalous Process: ~3000-7000 Nodes and ~9k-16k Edges

Introduction

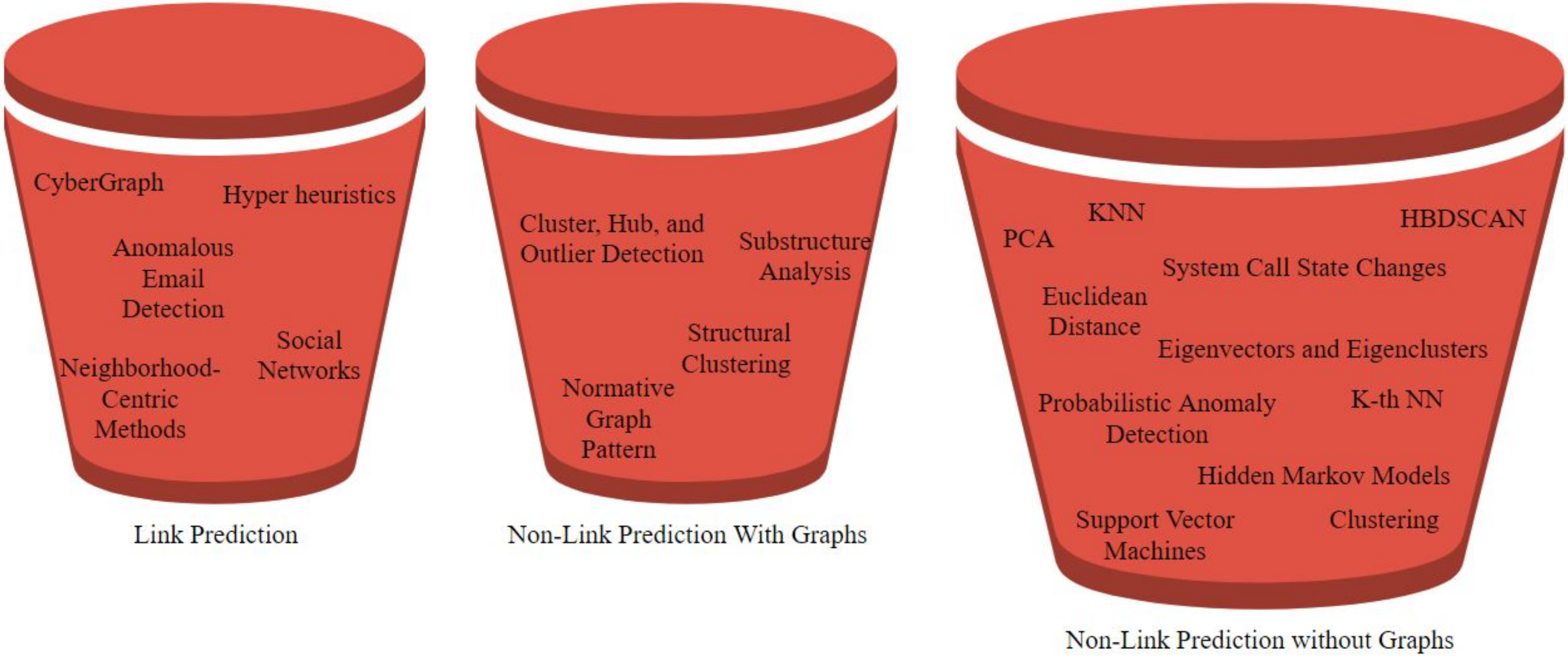
- MS in Cybersecurity in Computer Science
  - Machine Learning Course

- DARPA CHASE Research Assistantship
  - “The CHASE program seeks to develop automated tools to detect and characterize novel attack vectors, collect the right contextual data, and disseminate protective measures both within and across enterprises.”

GW SCHOOL OF ENGINEERING & APPLIED SCIENCE

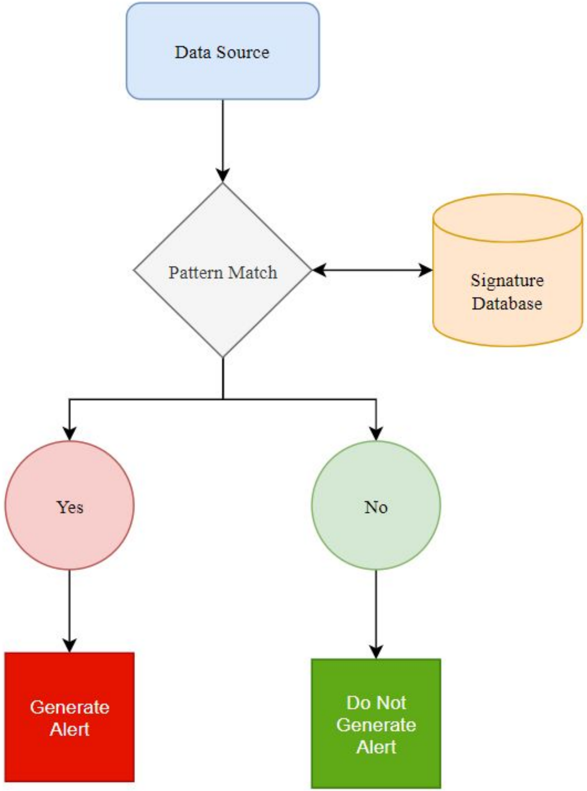
DARPA DEFENSE ADVANCED RESEARCH PROJECTS AGENCY

Prior Work in Anomaly Detection



Signature-based Detection Challenges

- Focus on pattern/signature matching in database
- Un-intelligent system
- Principal failure:
  - To detect unknown attacks
  - To detect patterns in behavior
- Common APT characteristics:
  - “Live off the land”
  - Zero-Days
  - Blend-in with the environment
  - Intelligent exploitation



Stage One: Link Prediction Threshold Optimization

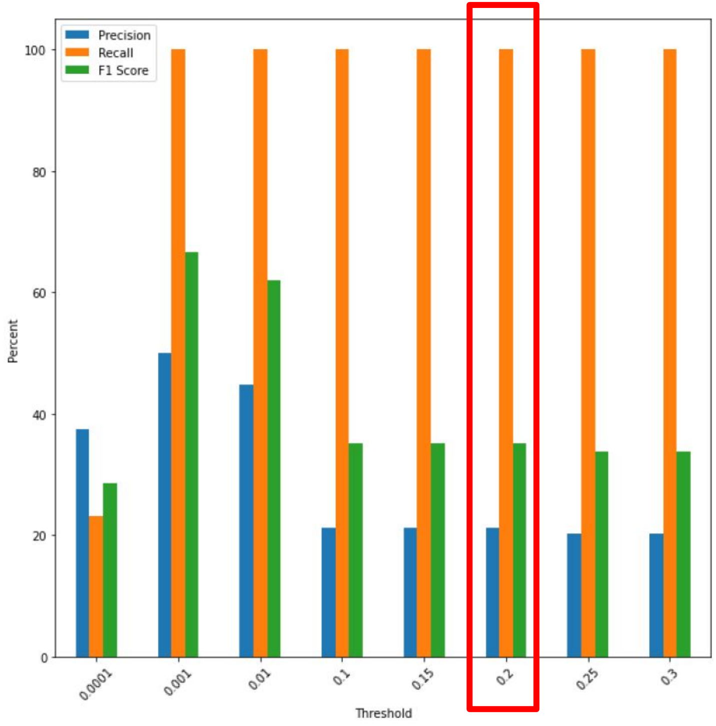


Figure 4.7: Host 0201 τ Evaluation

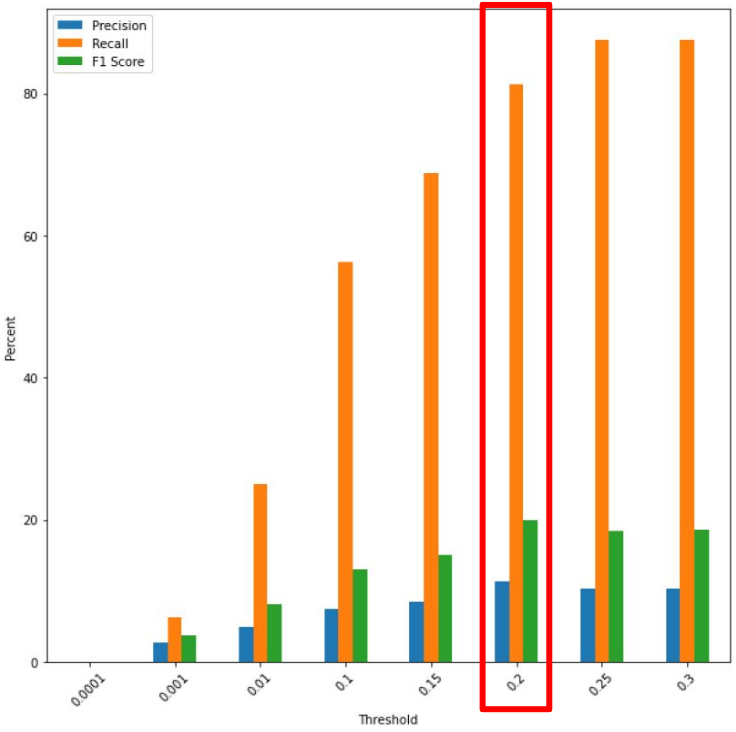


Figure 4.8: Host 0501 τ Evaluation

τ=0.20

P_e=0 P_e=1

Stage One: Random Walk Optimization

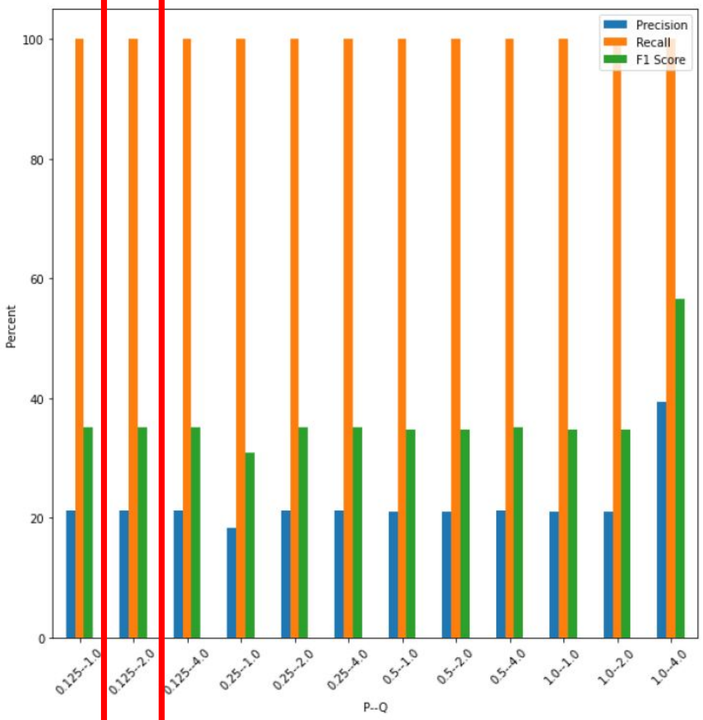


Figure 4.5: Host 0201 Random Walk Evaluation

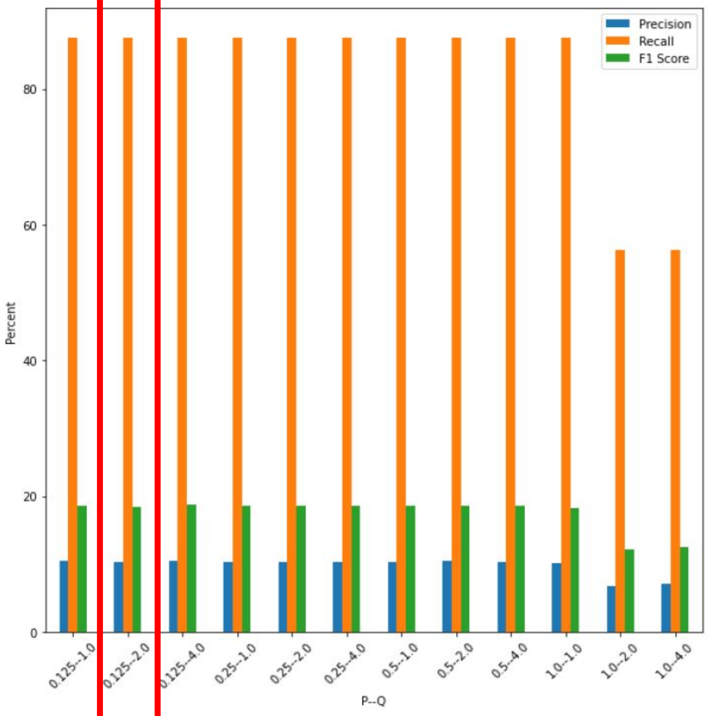
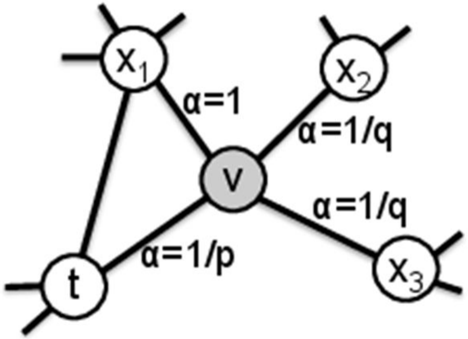
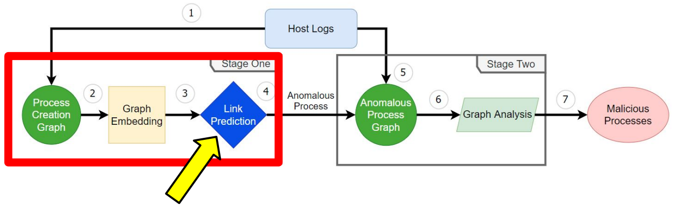


Figure 4.6: Host 0501 Random Walk Evaluation



Stage One Evaluation Metrics



- Metrics:
  - True Positive: edge **found in** Process Creation Graph that is present **in** Red Team notes
  - False Positive: edge **found in** Process Creation graph that is **not in** Red Team notes
  - False Negative: edge **not found in** Process Creation graph that is **in** Red Team notes
  - True Negative: edge **not found in** Process Creation graph that is **not in** Red team notes
- Important: Red Team notes do not track all Red Team activity

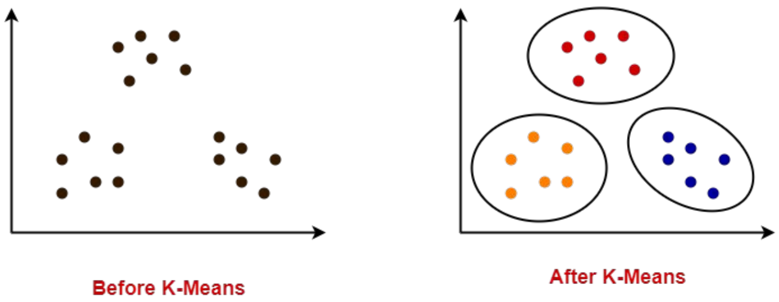
$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} = \frac{RedTeam_{edge}}{RedTeam_{edge} + RedTeam'_{edge}}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} = \frac{RedTeam_{edge}}{RedTeam_{edge} + RedTeam_{edge'}}$$

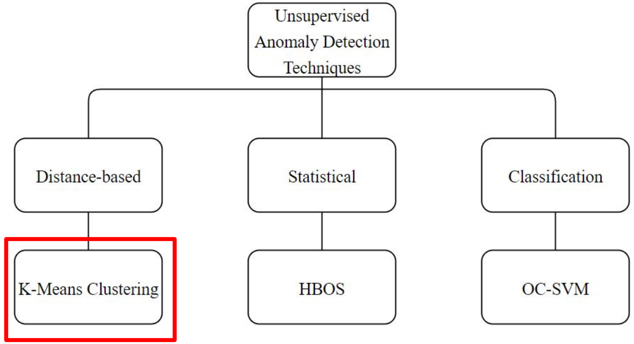
$$F1_{Score} = 2\frac{Recall * Precision}{Recall + Precision}$$

Algorithm Comparison: K-Means Clustering

- Algorithm
  - Iterative assignment of data points to clusters



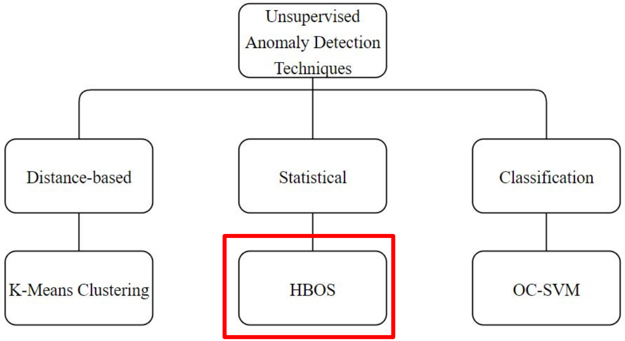
https://www.gatevidyalay.com/tag/k-means-clustering/



- Anomaly Definition:
  - Data with large distance to assigned centroid

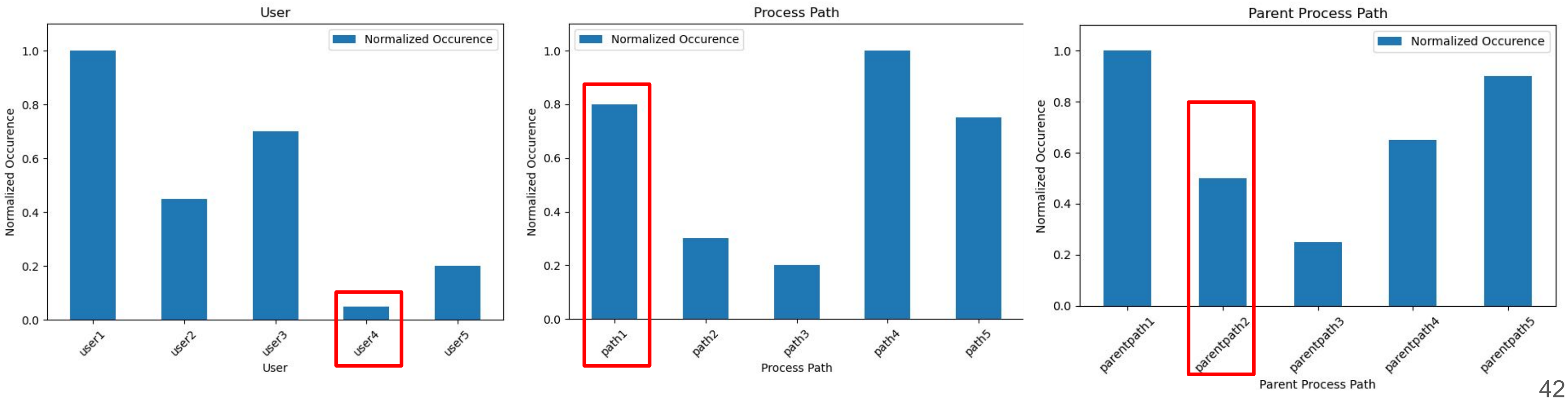
$$e_{anom} = \begin{cases} 1 & e_{euclid} \in \tau_d \\ 0 & \text{otherwise} \end{cases}$$

Algorithm Comparison: HBOS

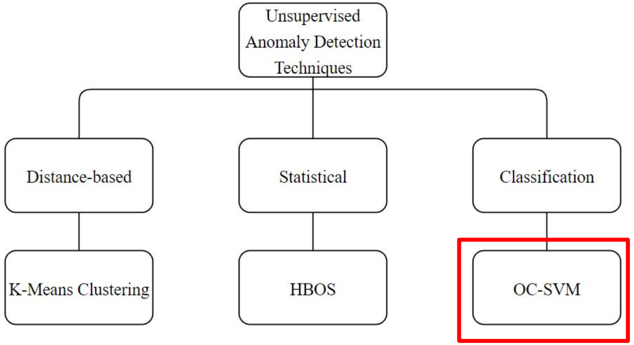


- Algorithm
  - Frequency calculation of features
  - Logarithmic sum of histograms creates score
- Anomaly Definition:
  - Highest scores represent the most anomalous activity

$$HBOS_{instance} = \sum_{i=0}^{d} \log \frac{1}{hist_i(instance)}$$

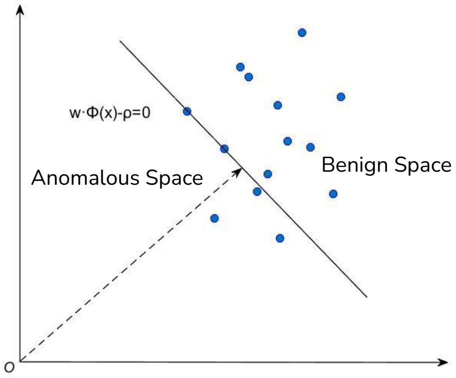


Algorithm Comparison: OC SVM



- Algorithm:
  - Learns a decision boundary to group input data

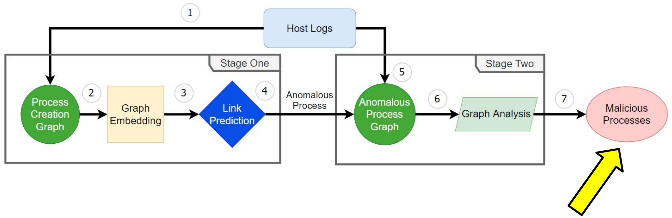
$$g(x) = \omega^T \phi(x) - \rho$$



- Anomaly Definition:
  - Any data point below the learned linear boundary is an anomaly
  - No threshold

$$label = \begin{cases} anomalous & \text{if } g(x) < 0 \\ benign & \text{if } g(x) > 0 \end{cases}$$

Example Results



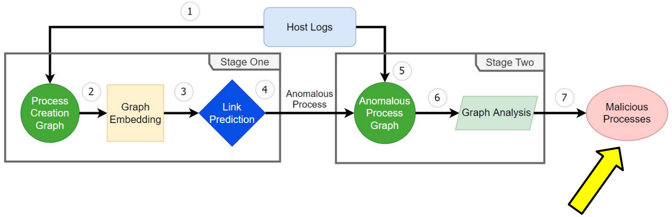
Host 0201

[('powershell.exe--SYSTEMIACOM\\zleazer--4632', **2252**),
('powershell.exe--SYSTEMIACOM\\zleazer--2952', **1632**),
('ping.exe--powershell.exe--2952', **1268**),
('explorer.exe--cmd.exe--2600', **383**),
('cmd.exe--SYSTEMIACOM\\zleazer--1284', **373**)]

Host 0501

[('backgroundtaskhost.exe--SYSTEMIACOM\\sysadmin--652', **3479**),
('powershell.exe--SYSTEMIACOM\\bantonio--2804', **2387**),
('powershell.exe--netstat.exe--1748', **2221**),
('powershell.exe--SYSTEMIACOM\\bantonio--648', **2041**),
('schtasks.exe--powershell.exe--648', **2041**)]

Results for Host 0201



- Placement of malicious activity at the highest percentiles of the results
    - Assumption 3 is affirmed
- Effectively filters false positives from Stage One
- Processes:
    - 4632, 2952, 1284
- Average Run Time: 30.966 sec

Threshold	Precision	Recall
Top 1	1.000	0.500
Top 5	0.800	0.800
Top 10	0.600	0.857
Top 15	0.467	1.000
Top 20	0.400	1.000

Table 4.8: Top-K Comparison for Host 0201

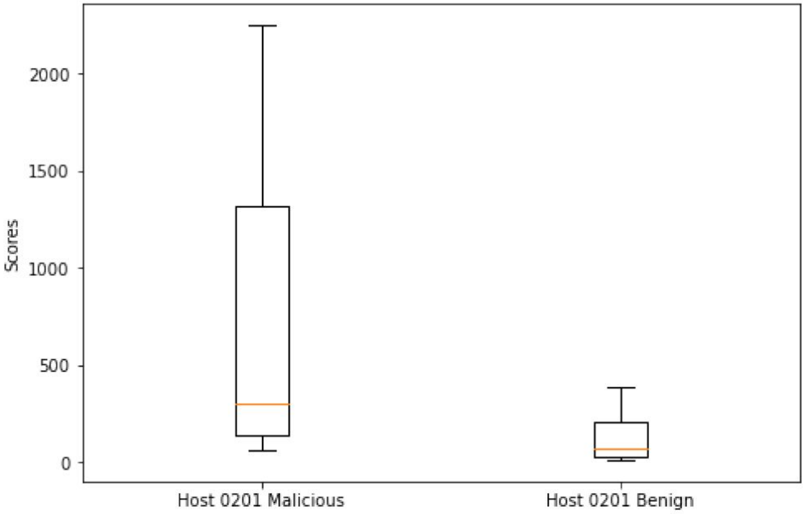
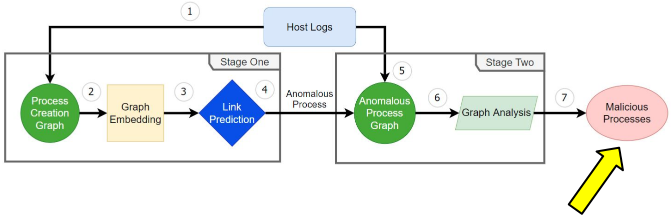


Figure 4.9: Box and Whisker Plot for Scores on Host 0201



Results for Host 0501

- Increased number of false positives impact results
- Placement of malicious activity at higher percentiles of results
    - Assumption 3 is affirmed
- Processes:
    - 2804 (5076), 1748, 648
- Average Run Time: 268.23 sec

Threshold	Precision	Recall
Top 1	0.000	0.000
Top 5	0.800	1.000
Top 10	0.400	1.000
Top 15	0.267	1.000
Top 20	0.200	1.000

Table 4.10: Top-K Comparison for Host 0501

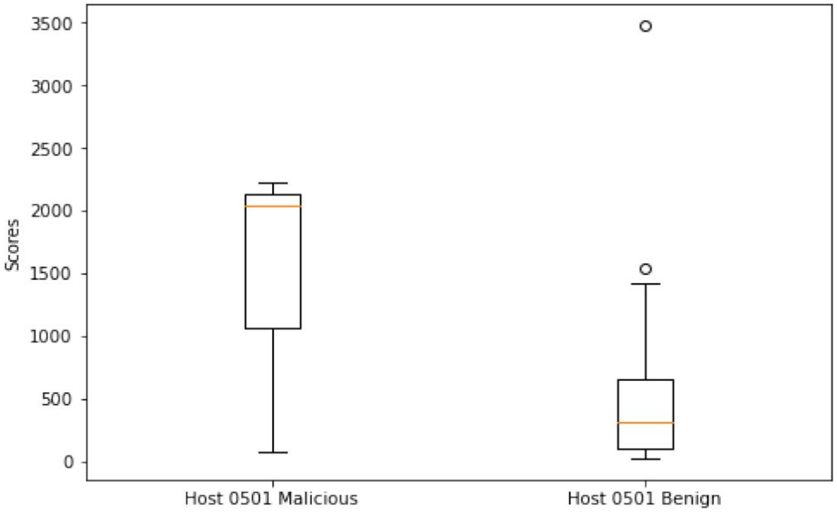


Figure 4.10: Box and Whisker Plot for Scores on Host 0501