NoDoze:Combating Threat Alert Fatigue with Automated Provenance Triage

Wajih Ul Hassan, Shengjian Guo, Ding Li, Zhengzhang Chen, Kangkook Jee, Zhichun Li, Adam Bates

26th Annual Network and Distributed System Security Symposium (NDSS) 2019

The Modern Cyber Threat Pandemic

Every company wants to keep their name off this chart

Source: World's Biggest Data Breaches, Information is Beautiful

Threat Detection

- Threat Detection Software (TDS) is the standard approach to security monitoring in large organizations.

- Even the most advanced tools are prone to **high false alert rates**

State of Threat Detection

Fireeye's "How Many Alerts is Too Many to Handle?" report:

Threat Alert Fatigue

A phenomenon when cyber analysts do not respond to threat alerts because they receive so many each day.

$ Waste an average of **$1.27** million every year

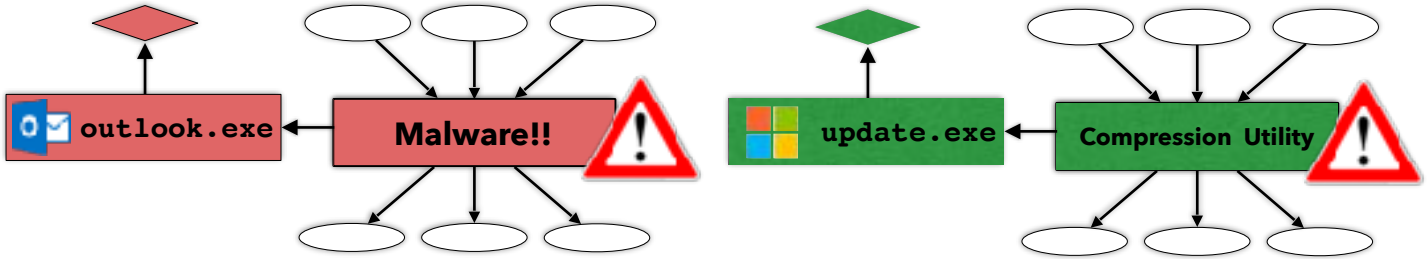
https://www2.fireeye.com/CMPG-IDC-Numbers-Game-Special-Report.html

Threat Alert Fatigue

Where are we going wrong?

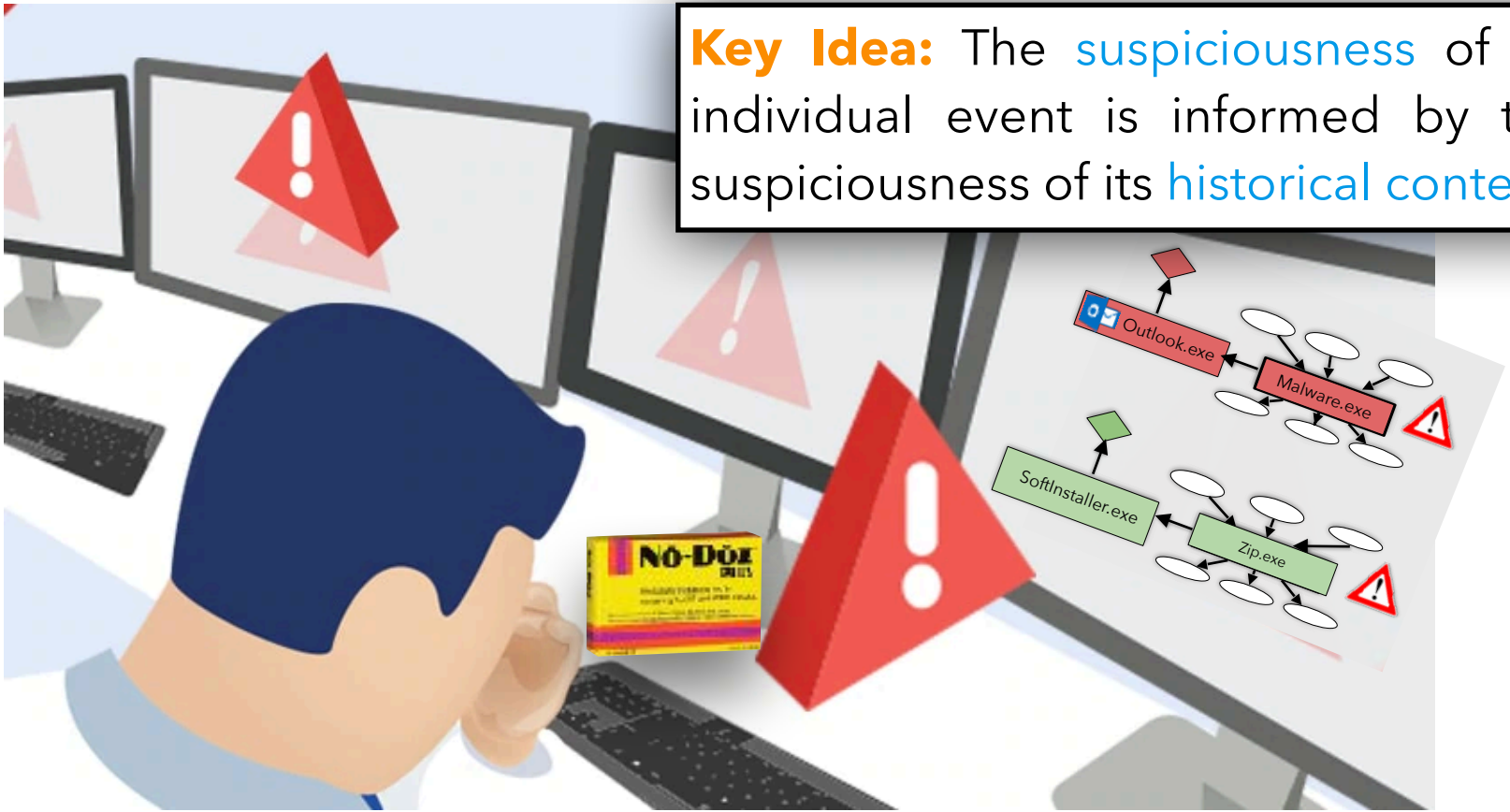
- Support for alert context is limited or non-existent
  - Alerts fire based on single-event rules
  - Rules are heuristic, curated by domain experts

Example rule: ALERT if process reads/writes many files in a short span of time



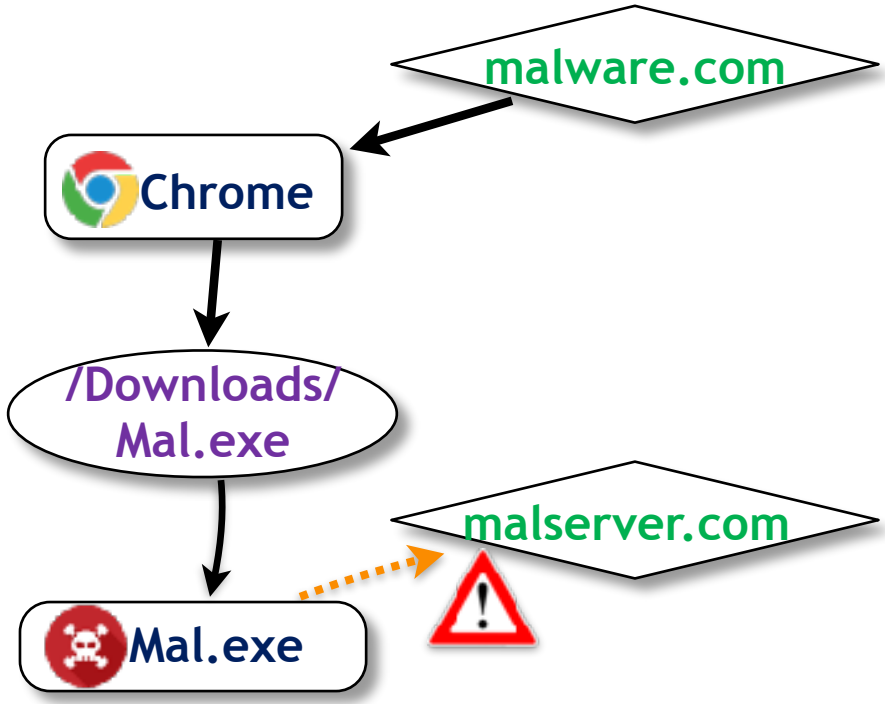
Combatting Alert Fatigue

Key Idea: The suspiciousness of an individual event is informed by the suspiciousness of its historical context.



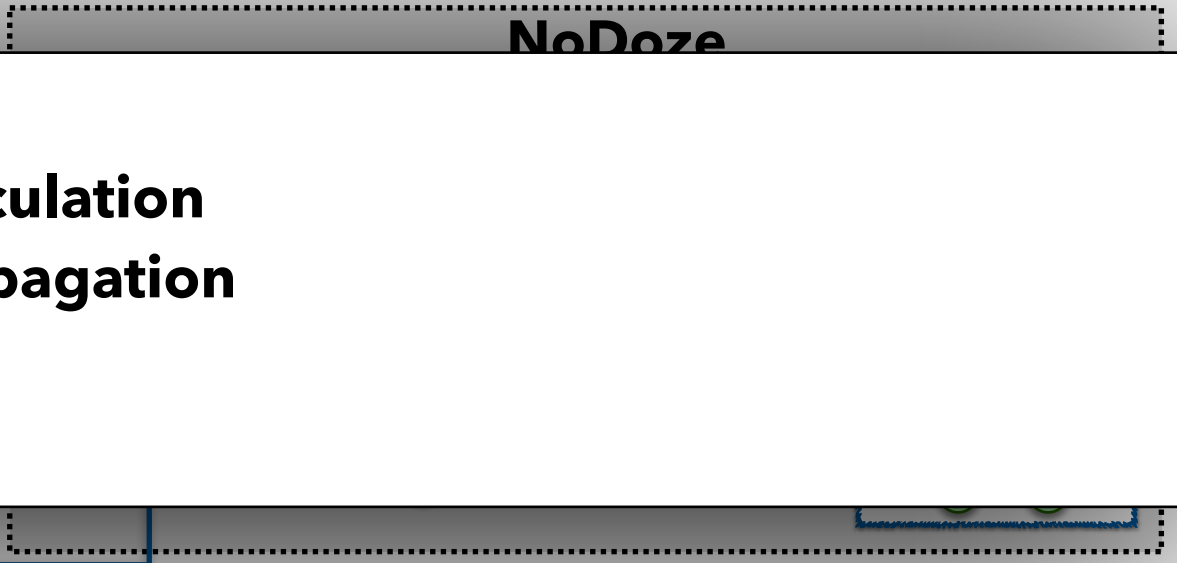
Threat Alert Investigation

- Life cycle of data object
  - Represented as graph
  - Vertex: File, Socket and Process
  - Edge: Causal dependency event
    - where each event E is a tuple of (SRC,DST,REL)
- Helpful in alert investigation
  - Querying root cause of the alert
  - Gives you context of the alert



Dependency Graph

NoDoze Workflow



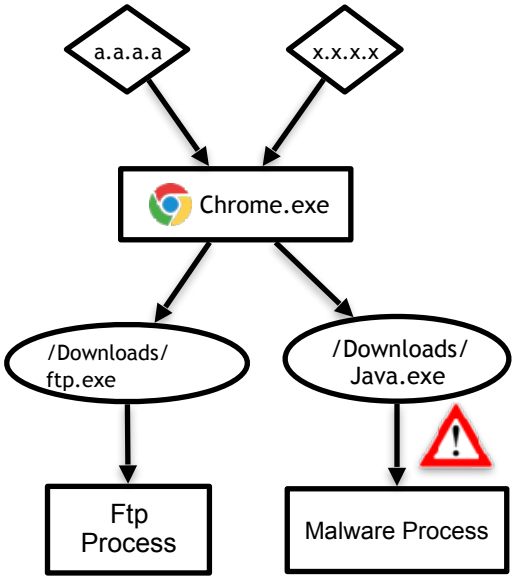
1. Anomaly Score Calculation
2. Anomaly Score Propagation
3. Graph Reduction

Anomaly Score Calculation

1. Use historic event data to build an **Event Frequency Database**
    - Encodes typical behavior within the organization

Anomaly Score Calculation

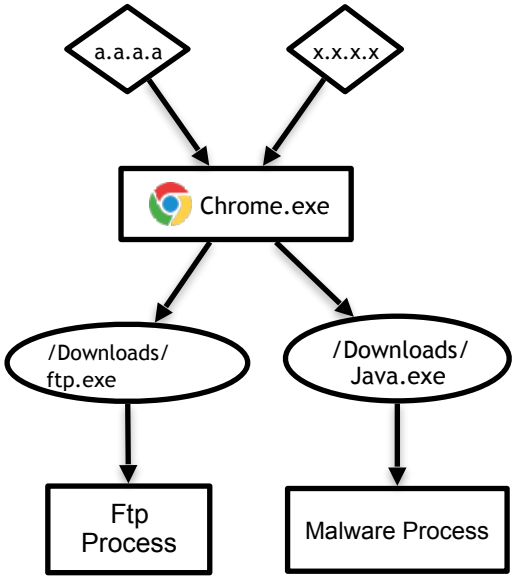
1. Use historic event data to build an **Event Frequency Database**
   - Encodes typical behavior within the organization
2. Generates provenance graph for each alert event.



Anomaly Score Calculation

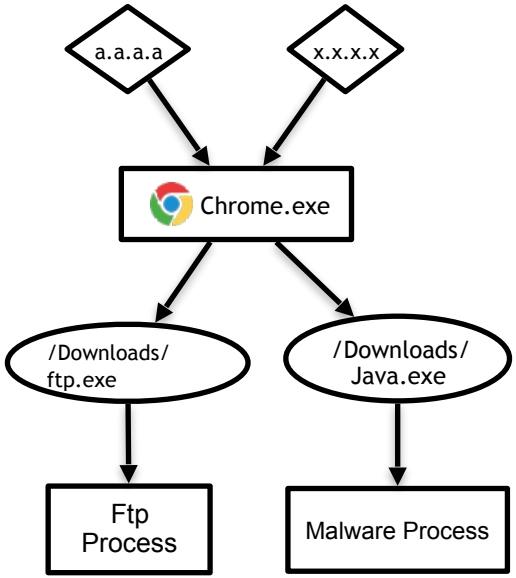
1. Use historic event data to build an **Event Frequency Database**
   - Encodes typical behavior within the organization
2. Generates provenance graph for each alert event.
3. Assign transition probability to each event (edge)
   - how often information flows from SRC to DST for particular REL

$$TransProbabilty(E) = \frac{Frequency(E)}{Frequency_{onlySRC}(E)}$$



Anomaly Score Calculation

1. Use historic event data to build an **Event Frequency Database**
   - Encodes typical behavior within the organization
2. Generates provenance graph for each alert event.
3. Assign transition probability to each event (edge)
   - how often information flows from SRC to DST for particular REL



How often does data flow from SRC to DST?

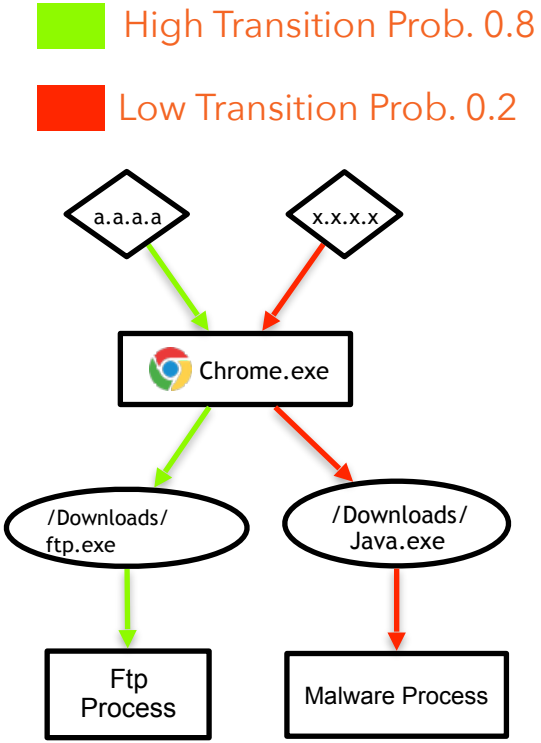
$$TransProbabilty(E) = \frac{Frequency(E)}{Frequency_{onlySRC}(E)}$$

How often does data flow from SRC to anywhere?

Anomaly Score Calculation

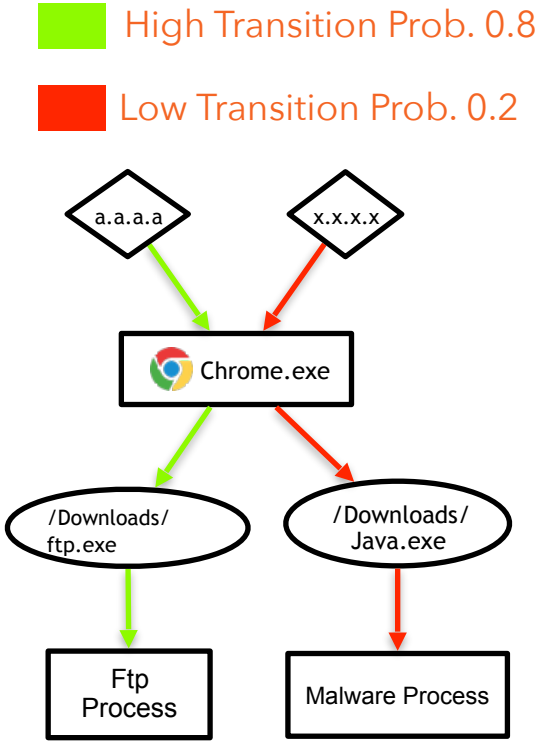
1. Use historic event data to build an **Event Frequency Database**
   - Encodes typical behavior within the organization
2. Generates provenance graph for each alert event.
3. Assign transition probability to each event (edge)
   - how often information flows from SRC to DST for particular REL

$$TransProbabilty(E) = \frac{Frequency(E)}{Frequency_{onlySRC}(E)}$$



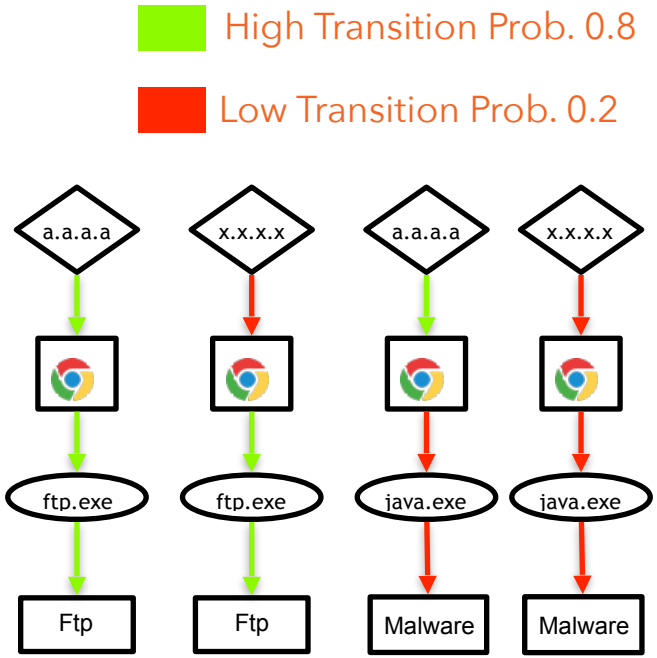
Anomaly Score Propagation

4. For Path $P = (E_1, E_2, \ldots, E_n)$ of length N in graph we calculate anomaly score as follows:



Anomaly Score Propagation

4. For Path $P = (E_1, E_2, \ldots, E_n)$ of length N in graph we calculate anomaly score as follows:



Anomaly Score Propagation

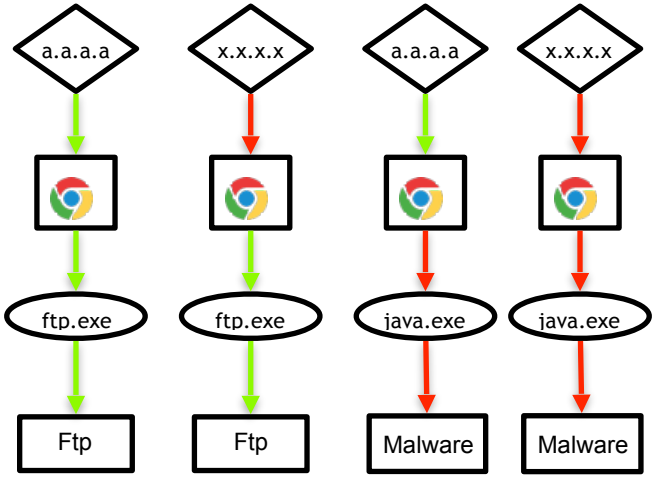
4. For Path $P = (E_1, E_2, \ldots, E_n)$ of length N in graph we calculate anomaly score as follows:

$$RegularityScore(P) = \prod_{i=1}^{N} IN(SRC_i) \times TransProb(E_i) \times OUT(DST_i)$$

IN/OUT scores account for total amount of data flowing in/out of the SRC and DST

High Transition Prob. 0.8

Low Transition Prob. 0.2



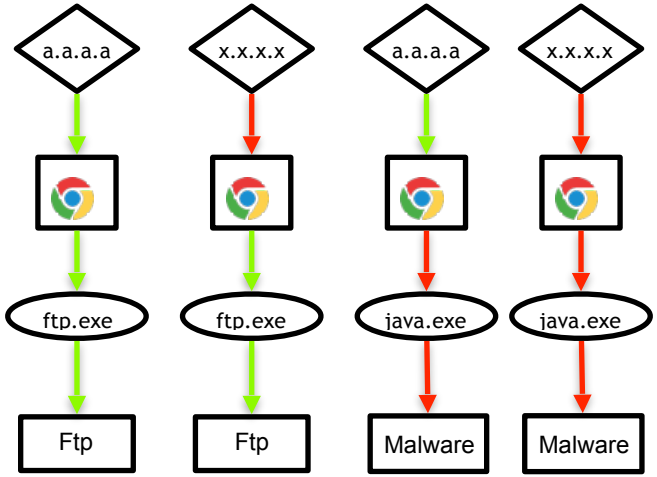
Anomaly Score Propagation

4. For Path $P = (E_1, E_2, \ldots, E_n)$ of length N in graph we calculate anomaly score as follows:

$$RegularityScore(P) = \prod_{i=1}^{N} IN(SRC_i) \times TransProb(E_i) \times OUT(DST_i)$$

High Transition Prob. 0.8

Low Transition Prob. 0.2



Regularity Scores = 0.512 0.128 0.032 0.008

For instance, IN and OUT score is 1.0 then:

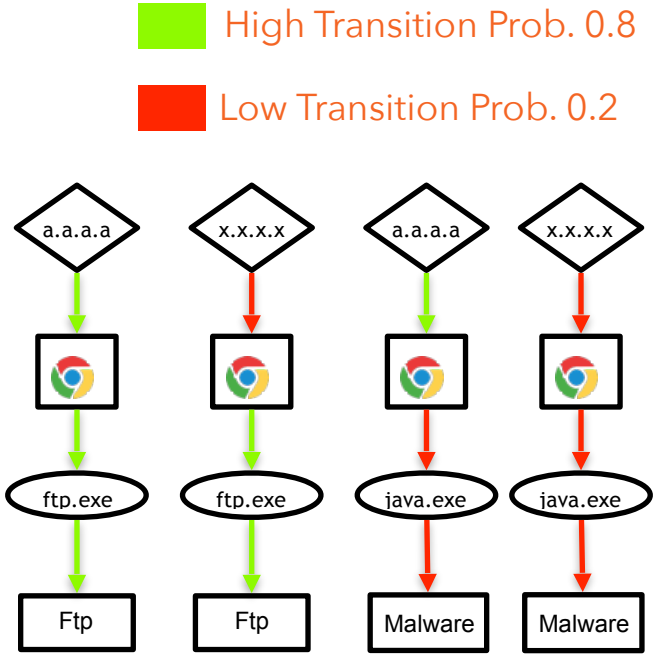
Anomaly Score Propagation

4. For Path $P = (E_1, E_2, \ldots, E_n)$ of length N in graph we calculate anomaly score as follows:

$$RegularityScore(P) = \prod_{i=1}^{N} IN(SRC_i) \times TransProb(E_i) \times OUT(DST_i)$$

$$AnomalyScore(P) = 1 - RegularityScore(P)$$

For instance, IN and OUT score is 1.0 then:



Regularity Scores =	0.512	0.128	0.032	0.008
Anomaly Scores =	0.488	0.872	0.968	0.992

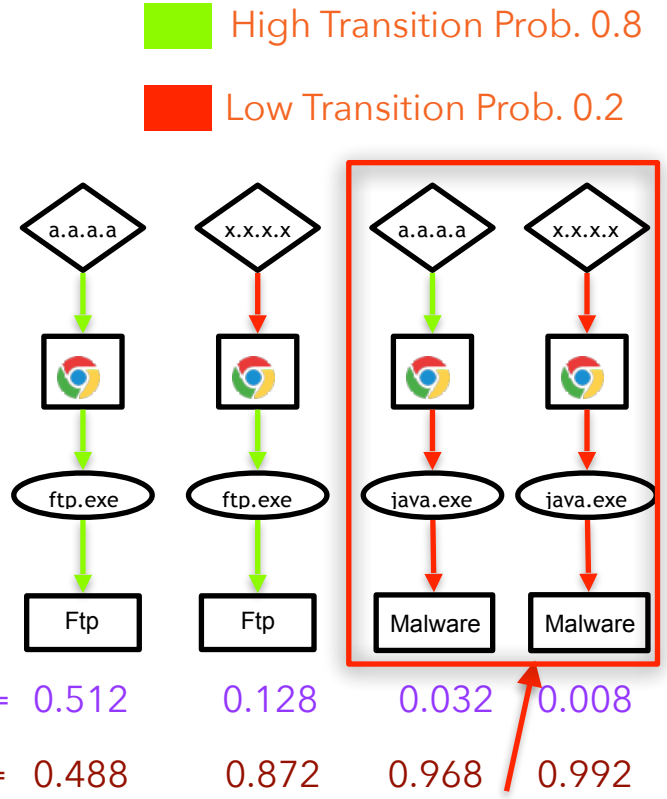
Anomaly Score Propagation

4. For Path $P = (E_1, E_2, \ldots, E_n)$ of length N in graph we calculate anomaly score as follows:

$$RegularityScore(P) = \prod_{i=1}^{N} IN(SRC_i) \times TransProb(E_i) \times OUT(DST_i)$$

$$AnomalyScore(P) = 1 - RegularityScore(P)$$

For instance, IN and OUT score is 1.0 then:



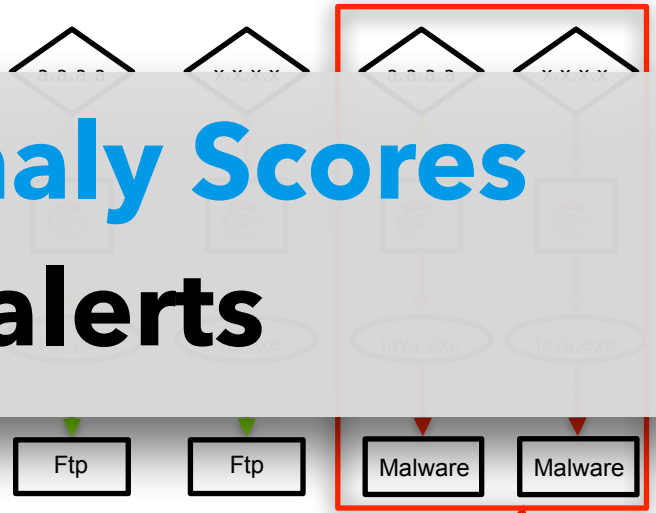
Anomaly Score Propagation

4. For Path $P = (E_1, E_2, \ldots, E_n)$ of length N in graph we calculate anomaly score as follows:

$$RegularityScore(P) = \prod_{i=1}^{N} IN(SRC_i) \times TransProb(E_i) \times OUT(DST_i)$$

$$AnomalyScore(P) = 1 - RegularityScore(P)$$

Use Aggregate Anomaly Scores to Triage threat alerts



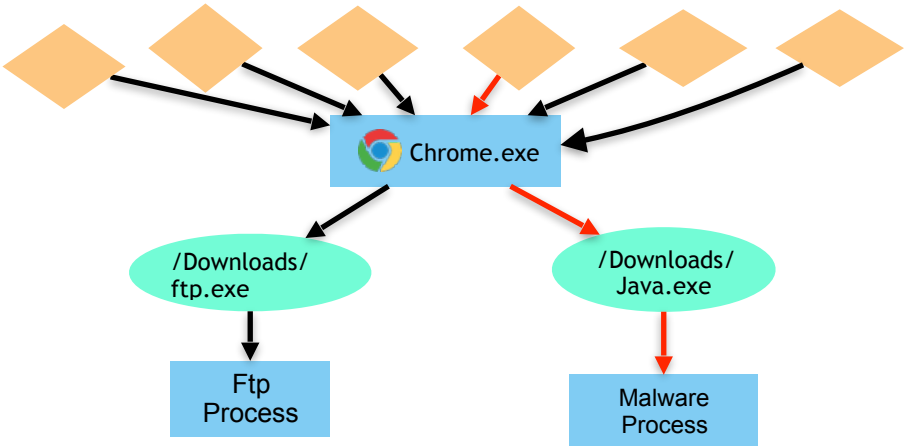
For instance, IN and OUT score is 1.0 then:

Regularity Scores =	0.512	0.128	0.032	0.008
Anomaly Scores =	0.488	0.872	0.968	0.992

Top 2 Anomalous Paths

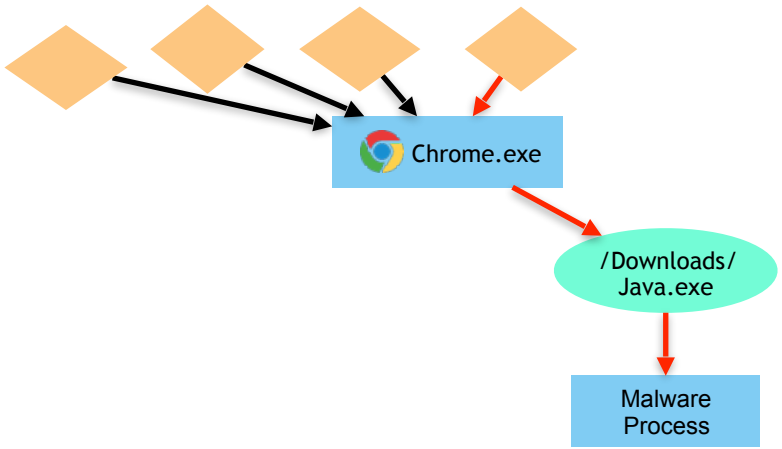
Graph Reduction

- A major issue in provenance analysis is **dependency explosion**
  - One output event depends on all input events that happen before it (the same process).



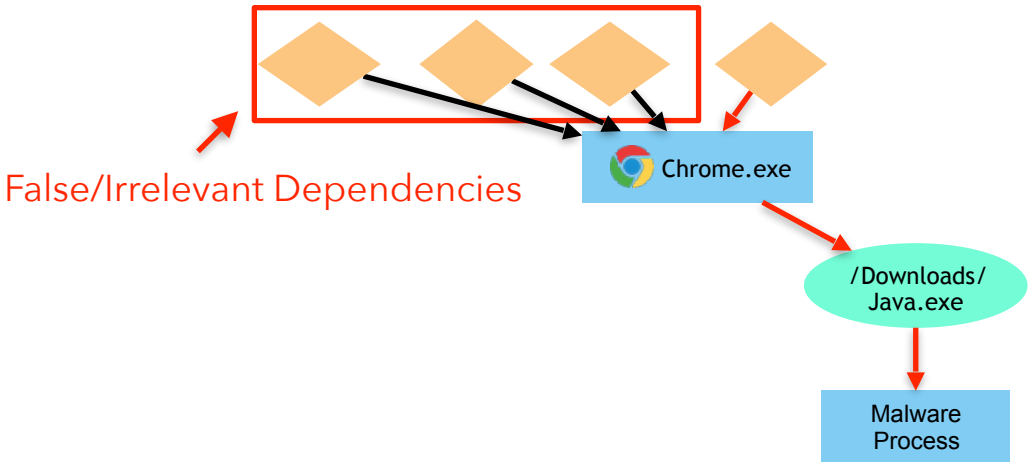
Graph Reduction

- A major issue in provenance analysis is **dependency explosion**
  - One output event depends on all input events that happen before it (the same process).



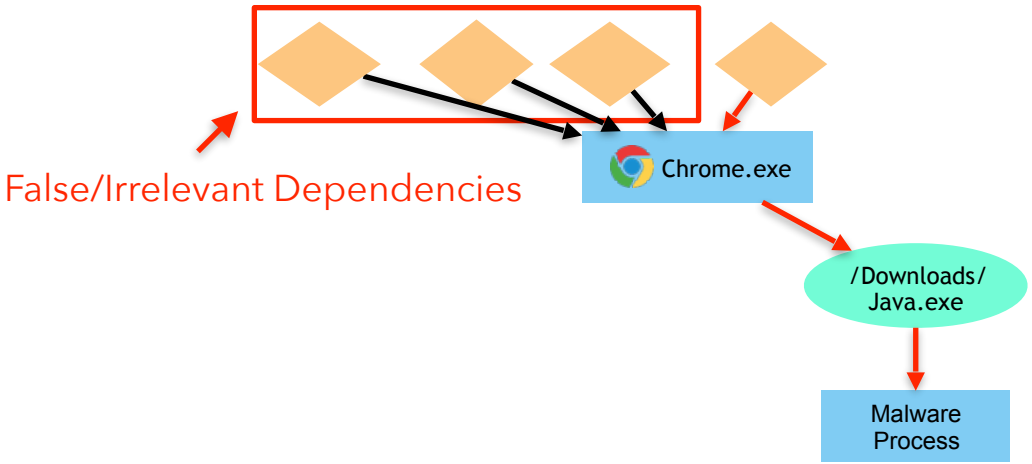
Graph Reduction

- A major issue in provenance analysis is **dependency explosion**
  - One output event depends on all input events that happen before it (the same process).



Graph Reduction

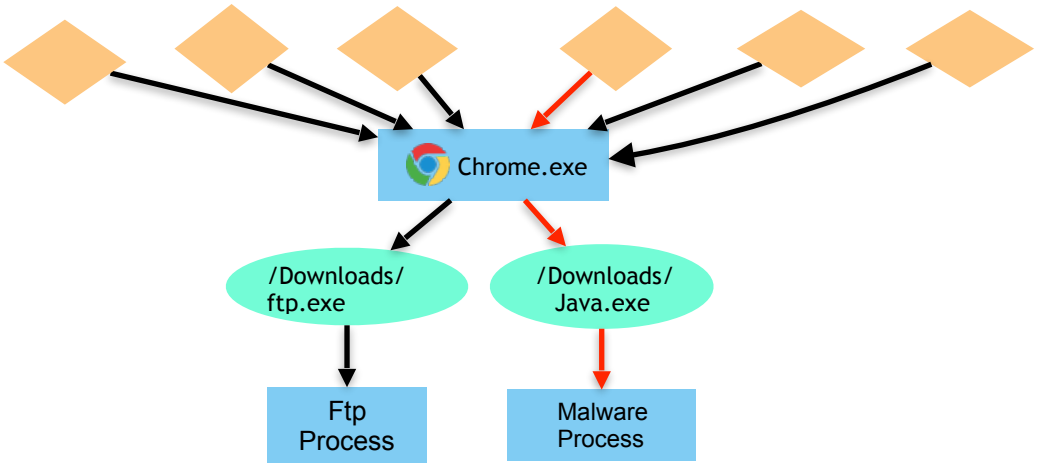
- A major issue in provenance analysis is **dependency explosion**
  - One output event depends on all input events that happen before it (the same process).



- Existing solutions require developer intervention

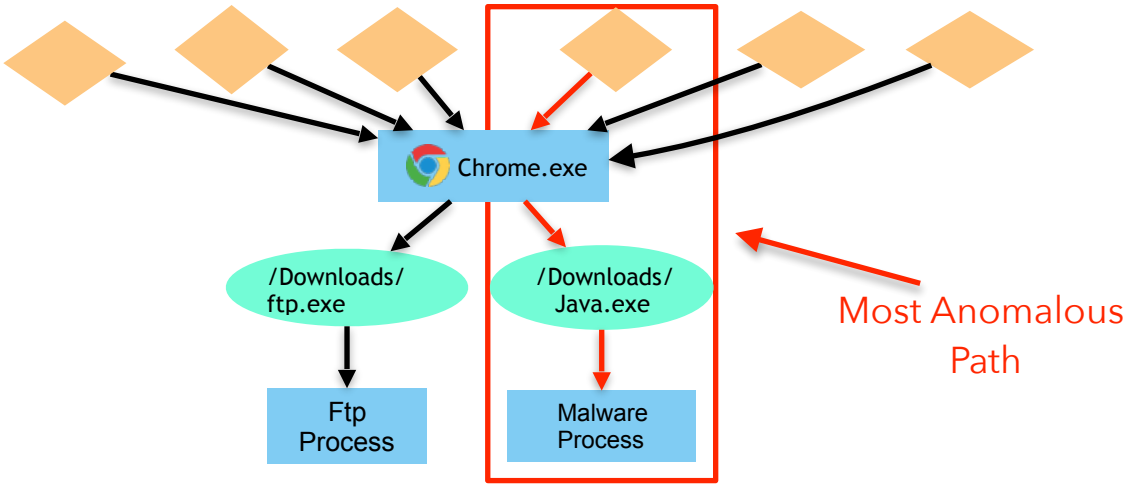
Graph Reduction

- NoDoze introduces **behavioral execution partitioning**
  - partition a program's execution between normal and anomalous behavior, prune normal paths.



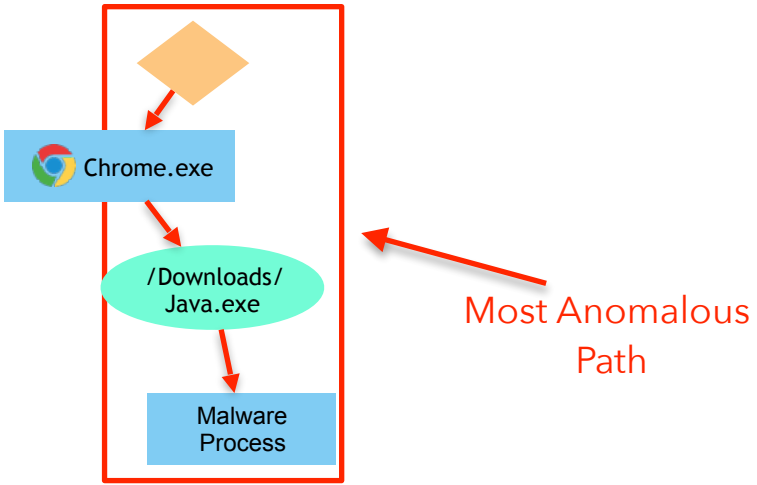
Graph Reduction

- NoDoze introduces **behavioral execution partitioning**
  - partition a program's execution between normal and anomalous behavior, prune normal paths.



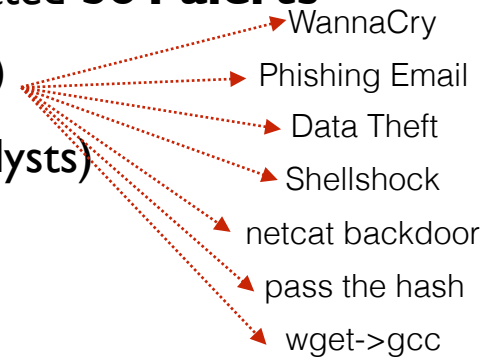
Graph Reduction

- NoDoze introduces **behavioral execution partitioning**
  - partition a program's execution between normal and anomalous behavior, prune normal paths.



NoDoze Evaluation

- Experimentally validated at NEC Labs using their commercially-available threat detection software (NEC ASI System).
- Provenance data from **190 hosts** (heterogenous network)
- Event Frequency Database populated with **1 month** data
- Evaluation engagement took place over **5 days**
- *Underlying Threat Detection Software* generated **364 alerts**
  - **50 True Alerts** (we injected these)
  - **314 False Alerts** (validated by analysts)



COMPUTER SECURITY
NEC LABORATORIES

Summary of Results

84%
reduction in false alarms

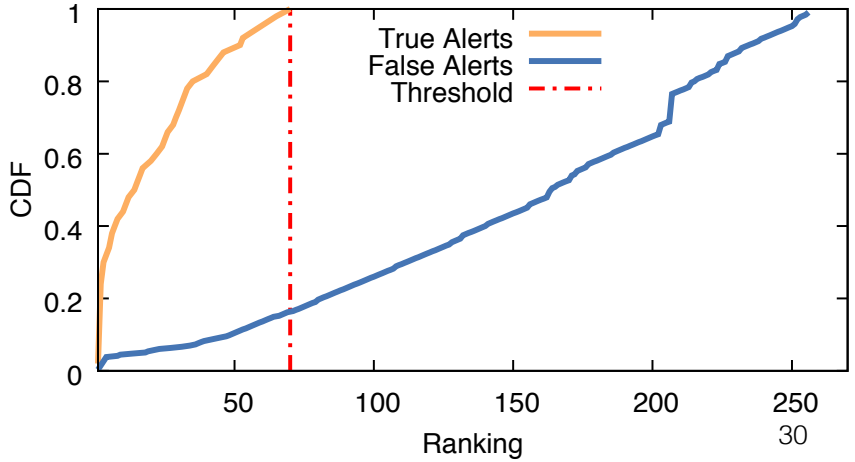
>90
employee-hours saved

2
orders smaller graph

Threat Alert Triage

84% reduction

- To prioritize alerts, just sort by anomaly score!
- Can we go further? **Yes**
  - If there is major separation between scores of True Alerts and False Alerts, we can set a separation threshold for alerts that fall beneath a certain score.
    - Threshold can be set experimentally by analysts based on past investigations.

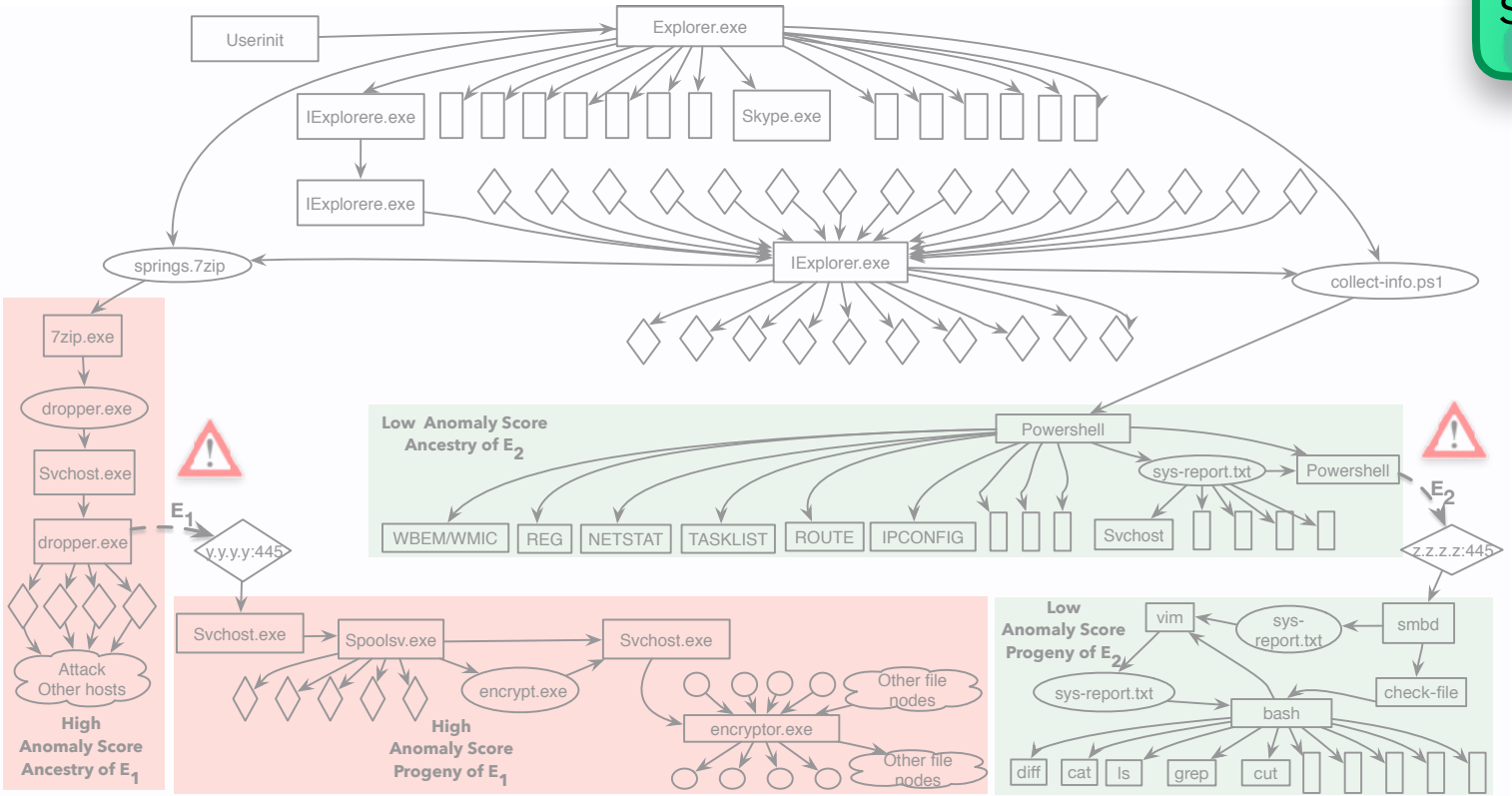


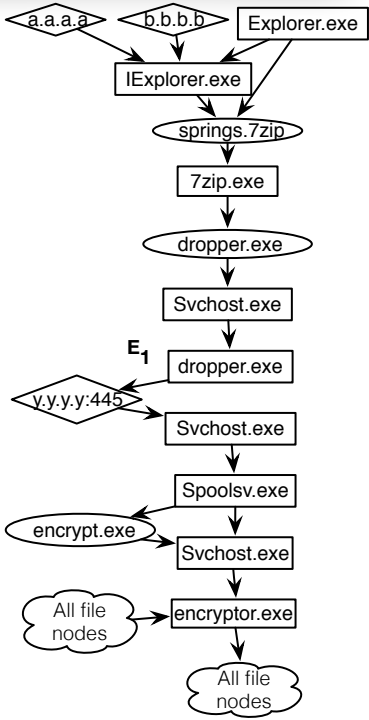
Time Saved

- Studies have shown that it takes **20+ mins** on average to investigate each alert
- In our dataset we have total 314 false alerts collected from underlying threat detection software
  - Take 104 hours to investigate
- NoDoze reduces 84% of 314 false alerts
  - Saved more than **90** hours

Graph Reduction

2 orders smaller graph





Conclusion

- We develop NoDoze – a threat alert triage and investigation system
- It leverages historical information and contextual alerting to improve state-of-the-art threat detection softwares
- Evaluation results show that our system substantially reduces the slog of investigating false alarms

Conclusion

- We develop NoDoze – a threat alert triage and investigation system
- It leverages historical information and contextual alerting to improve state-of-the-art threat detection softwares
- Evaluation results show that our system substantially reduces the slog of investigating false alarms

Thanks & Questions

whassan3@illinois.edu

Backup slides

Why we need TDS?

- Using NoDoze as a TDS is prohibitively costly
  - Graph analysis on every event happening in enterprise
- Lot of research to curate these rules
  - Efficiently generate threat alerts
  - Use these alerts as a starting point

What about False negative

- Two reasons to miss attacks:
  - Underlying TDS miss attacks
  - NoDoze separation threshold is two low
- Goal of NoDoze is to triage
- Separation Threshold is configurable
  - Based on organization setup such as num. of hosts and workload

Anomaly Score Normalization

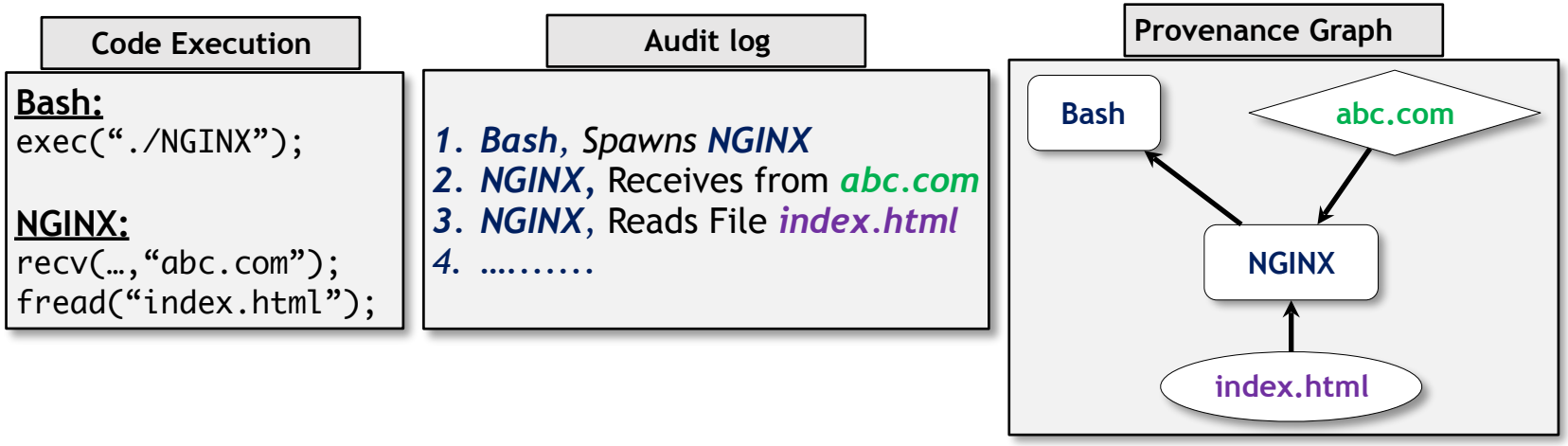
$$AnomalyScore(P) = 1 - \prod_{i=1}^{N} IN(SRC_i) \times TransProb(E_i) \times OUT(DST_i)$$

Normalize the path scores

- Longer paths tends to have higher score in above equation
- Remove scoring bias by calculating decay factor using random sampling approach

Data Provenance aka Audit log

- Lineage of system activities
- Represented as Graph
  - Vertex: File, Socket and Process
  - Edge: Causal dependency event



Linux Auditd Architecture