MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Sometimes, You Aren't What You Do: Mimicry Attacks against Provenance Graph Host Intrusion Detection Systems

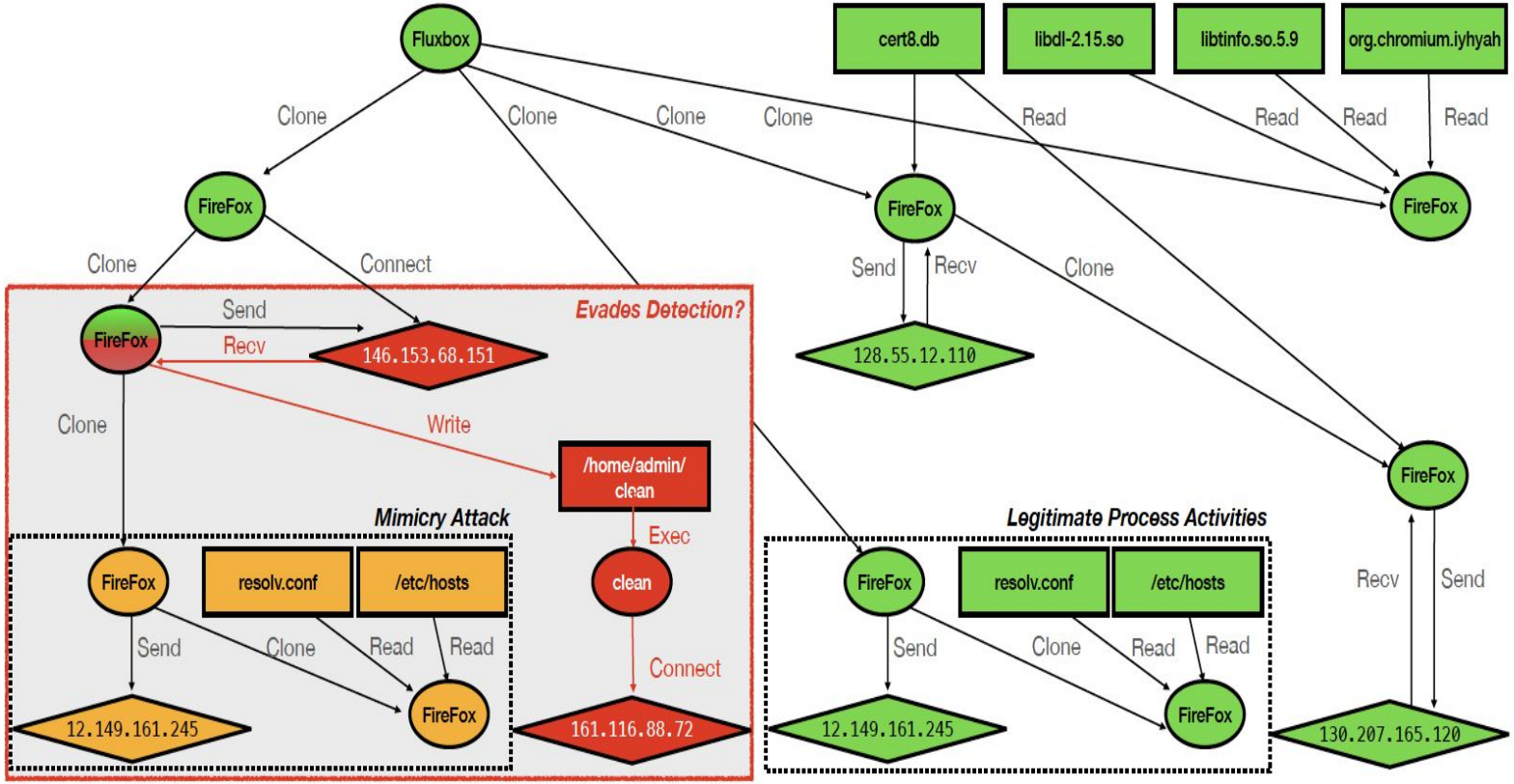
Akul Goyal, Xueyuan Han, Gang Wang, Adam Bates

Presented by: Michael Hu

Provenance Graphs

- We define a provenance graph as G = (V,E), where V = $\{v_j\}^{|V|}_{j=1}$ is a set of vertices and E = $\{e_j\}^{|E|}_{j=1}$ is a set of edges

- Each vertex V represents an entity that was accessed in the system
  - Includes files, processes, network sockets, etc
- Each edge E represents an a specific system event that was observed


Fluxbox
cert8.db
libdl-2.15.so
libtinfo.so.5.9
org.chromium.iyhyah
Clone
Clone
Clone
Clone
Read
Read
Read
Read
FireFox
FireFox
FireFox
Clone
Connect
Send
Recv
Clone
Evades Detection?
Send
Recv
146.153.68.151
128.55.12.110
Clone
Write
/home/admin/
clean
FireFox
Mimicry Attack
Legitimate Process Activities
Exec
clean
FireFox
resolv.conf
/etc/hosts
FireFox
resolv.conf
/etc/hosts
Recv
Send
Send
Clone
Read
Read
Send
Clone
Read
Read
Connect
12.149.161.245
FireFox
161.116.88.72
12.149.161.245
FireFox
130.207.165.120

Prov-HIDS

- Let each graph G_i in a dataset G be associated with a label $y_i \in \{0, 1\}$, where 0 is benign and 1 is malicious, such that $G = \{G_i, y_i\}^{|G|}_{i=1}$
- The objective of a Prov-HIDS $f(G_i)$ is to minimize the loss function L:

$$\mathcal{L} = 1 - \frac{1}{|\mathcal{G}|} \sum_{i=1}^{|\mathcal{G}|} \mathbb{1}(f(G_i) = y_i)$$

where $\mathbb{1}(x) = 1$ if x is true; otherwise, $\mathbb{1}(x) = 0$.

Prov-HIDS Cont.

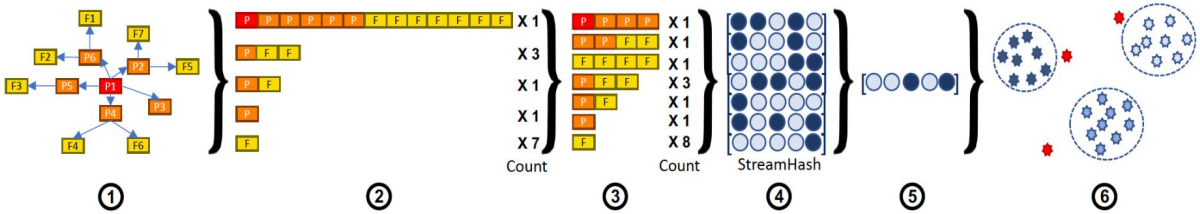
- The Prov-HIDS $f(G_i)$ makes a classification decision by encoding substructures within the graph Gi and then compares them to the substructures of a pre-encoded set of benign graphs.

- An anomaly is raised if Gi's substructures deviate significantly from the known benign substructures:

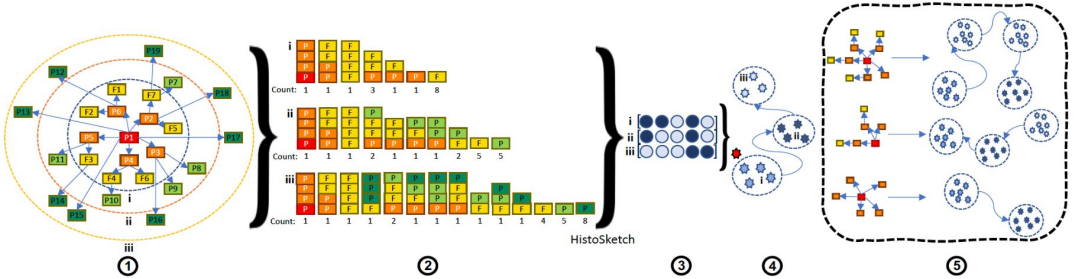
$$f(G_i) = \mathbb{1}(\mathcal{F}^{\delta}(\mathcal{E}^{\kappa}_{\lambda}(\mathcal{N}^{\gamma}_{\beta}(G_i))) \geq \alpha)$$

- $N^{\gamma}_{\beta}(G_i)$ is a deconstruction function that returns a set of substructures
- $E^{\kappa}_{\lambda}(Z_i)$ is an encoding function that summarizes Z_i into an L-Dimensional vector V_i
- $F^{\delta}(V_i)$ is a distance function that compares V_i against a set of learned graph encodings and returns the smallest distance between a graph in δ and V_i
- α is a distance threshold, under which G_i is considered benign.

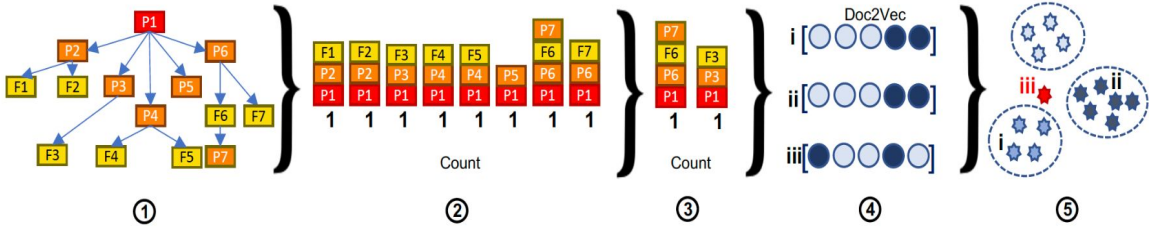
StreamSpot



Unicorn



ProvDetector



Types of Prov systems

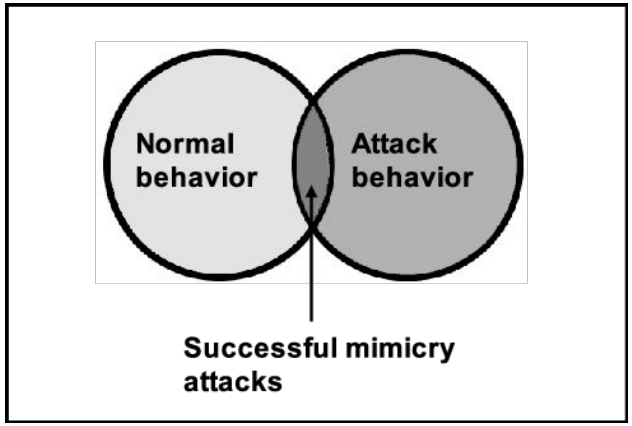
System	Function	Method	Learning Task	Code?
StreamSpot [17]	Detect.	Unsup.	Neigh.-based Whole Graph	✔
FRAPpuccino [25]	Detect.	Unsup.	Neigh.-based Whole Graph	✔
Unicorn [18]	Detect.	Unsup.	Neigh.-based Whole Graph	✔
Pagoda [20]	Detect.	Unsup.	Path-based Whole Graph	✔
P-Gaussian [26]	Detect.	Unsup.	Path-based Whole Graph	✖
ProvDetector [19]	Detect.	Unsup.	Path-based Subgraph	✖
PIDAS [27]	Detect.	Unsup.	Path-based Subgraph	✖
SIGL [21]	Detect.	Unsup.	Whole Graph Autoencoder	✖
Hercule [28]	Invest.	Sup.	Log Community Detection	✖
ATLAS [29]	Invest.	Sup.	Log Sequence Modeling	✔
NoDoze [30]	Invest.	Unsup.	Historic Event Analysis	✖
Holmes [31]	Invest.	Unsup.	Historic Event Analysis	✖

The BIG question

Are malicious acts innately distinct from regular activity?

What is a Mimicry Attack?

- Definition:
  - An attack where attackers pattern their actions such that they are indistinguishable from benign processes.



Threat Model

- Attacker Capabilities
  - Sophisticated APT with techniques and objectives that are consistent with the MITRE ATT&CK knowledge base
- Masquerading
  - Have the ability to use any tools to disguise themselves as legitimate security tools
- Gathering of Host Information
  - Adversaries will use reconnaissance to aid other attack operations
  - Attacker has access to procedures that enable them to access or infer the contents of audit logs

Observations of Prov-HIDS

- Adversaries can exert influence over an attack's embedding

- Prov-HIDS takes the entire systems executions as input, as such an attacker can arbitrarily introduce behavior that confuses the classification

Observations of Prov-HIDS Cont.

- Adversarial additions to an attack's embedding can be made indistinguishable from benign behavior
- During vectorization, deconstruction disassociates neighborhoods through bounded branching and depth.
- Therefore, the behavior's embedded representation will map to a benign behavior if it is more than β hops away from the root of the attack graph.

Evasion Tactics/Gadgets

- Abusing Unweighted Graph Encoding
- Abusing Distributional Graph Encoding
- Abusing Downsampled Graph Encoding

Abusing Unweighted Graph Encoding

- When a Prov-HIDS summarizes a graph in such a way that every substructure is equally weighted, an attacker could change the graphs embedding simply by adding additional activity

Steps

- Attacker starts by profiling the target system to identify a large number of graph substructures associated with benign activity

- Select a batch of benign substructures and replicate the system activities

Abusing Distributional Graph Encoding

- Some Prov-HIDS use techniques that preserve the underlying distribution by weighing each unique substructure according to its prevalence

- Idea is that there should be a significant difference between the distribution of substructures in malicious graphs vs benign graphs

Steps

- Attacker profiles the target system to identify the relative frequency of each observed substructure associated with benign activity

- Select a batch of benign substructures that preserves the distribution and replicate the system activities that produce those substructures

Abusing Downsampled Graph Encoding

- Some systems use downsampling as a technique, it improves the efficiency of training and generalizability of the model
- By making decisions about which subgraphs are relevant to the classification decision, DS(·) acts as the decision function.
- This causes problems as DS works over the entire graph, while the core decision function F is over the embedding space

Steps

- Profile the target system while monitoring the behavior of DS(·) on different observed sequences

- Select a batch of benign substructure sequences of parameterizable size that bypass the downsampling operation

What Tactics Work on What Models

- Abusing Unweighted Graph Encoding
  - StreamSpot
  - Pagoda
  - FRAPpuccino
  - PIDAS
- Abusing Distributional Graph Encoding
  - Unicorn
  - Pagoda
  - SIGL
  - P-Gaussian
- Abusing Downsampled Graph Encoding
  - ProvDetector

Procedures and Evaluation

Research Questions

- How effective are our mimicry gadgets against the five state-of-the-art, exemplar Prov-HIDS, StreamSpot, Unicorn, ProvDetector, Pagoda, and a SIGL-like full graph autoencoder?
- Are our mimicry gadgets interoperable and generally applicable to other detection mechanisms Relatedly, how much knowledge does the attacker need when using our gadgets to successfully evade detection?
- Can the attacker practically deploy our mimicry gadgets in the real world?
- How does our evasion strategies compare to previous, domain-general graph evasion techniques?
- What is the runtime p[illegible]e of our approach?

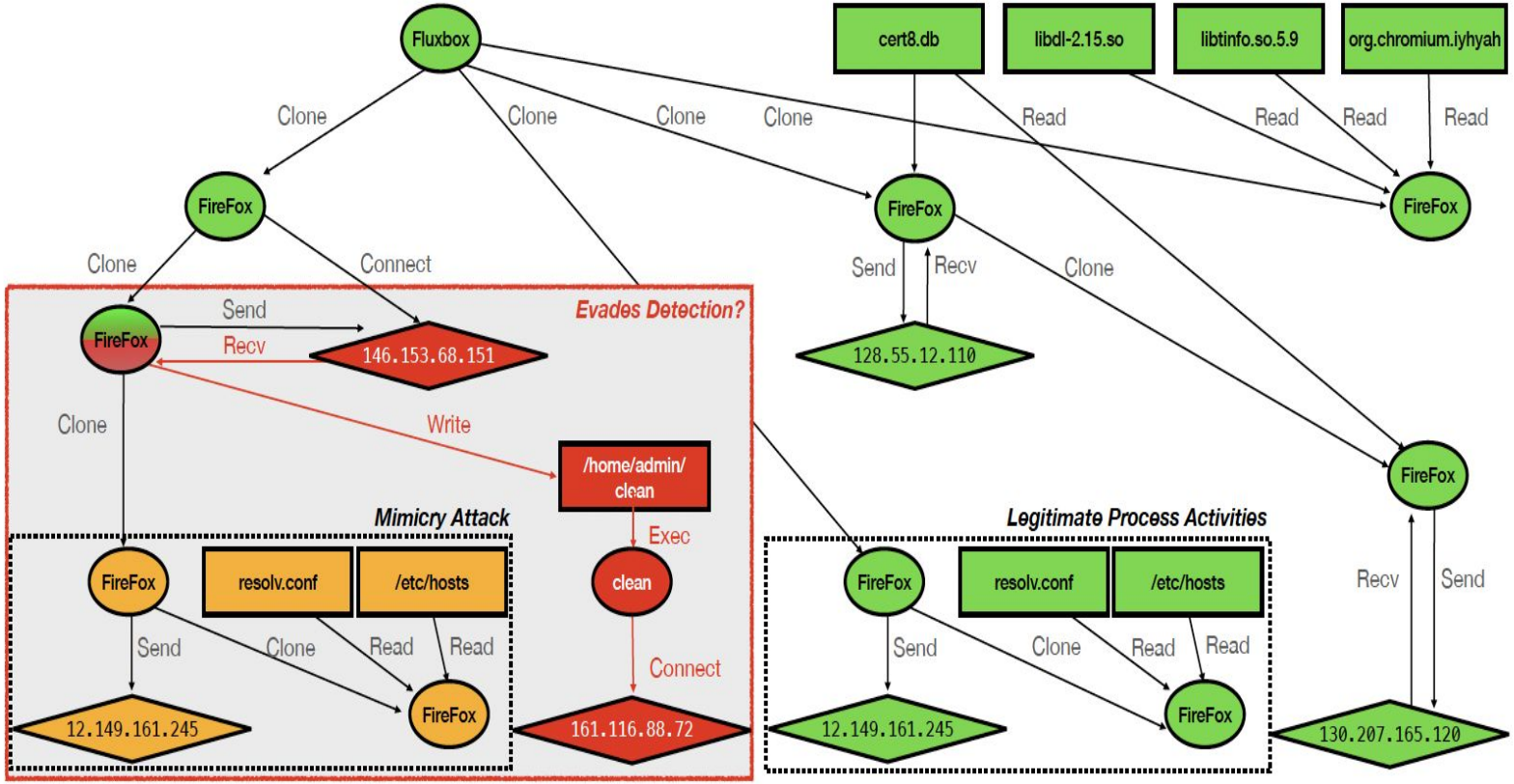
Datasets

- StreamSpot

- DARPA Transparent Computing Engagement 3

Attack

- User clicks a malicious URL that triggers a bug in Flash that enables root access
- Exploits a backdoor in Firefox and injects a binary executable
- Installs “Drakon” in its process memory and Drakon subsequently spawns a new process with root privileges that connects to the attacker’s server, giving the attacker full access


Fluxbox
cert8.db
libdl-2.15.so
libtinfo.so.5.9
org.chromium.iyhyah
Clone
Clone
Clone
Clone
Read
Read
Read
Read
FireFox
FireFox
FireFox
Clone
Connect
Send
Recv
Clone
Evades Detection?
Send
Recv
146.153.68.151
128.55.12.110
Clone
Write
/home/admin/
clean
Mimicry Attack
Legitimate Process Activities
Exec
clean
FireFox
resolv.conf
/etc/hosts
FireFox
resolv.conf
/etc/hosts
Send
Clone
Read
Read
Send
Clone
Read
Read
Connect
12.149.161.245
FireFox
161.116.88.72
12.149.161.245
FireFox
Recv
Send
130.207.165.120

StreamSpot

- Consists of 5 benign scenarios and 1 attack
  - Note that the training was done only on browsing CNN scenario to give advantage to the detection system
- Each behavior was recorded 100 times, resulting in 600 graphs
  - Each benign graph consisted of 295k edges
  - Each attack graph consisted of 28k edges
- Although seems limited and not realistic, this is valuable because the attack graphs consist only attack behavior

DARPA Transparent Computing Engagement 3

- Describes a professional red team's attempts to penetrate a small network of hosts.
- Includes a single provenance graph broken up into 25 different time periods
- The total activity in this dataset is approximately 4.8M edges
- More realistic system activity and features attack behavior occurring alongside normal system activity.

Gadget Implementation and Evasion Procedure

- Gadgets were written in Python

- Inputs:
  - The attack graph that is to be modified
  - A providence graph of benign activity
  - Number of edges/substructures to add
  - A point of insertion

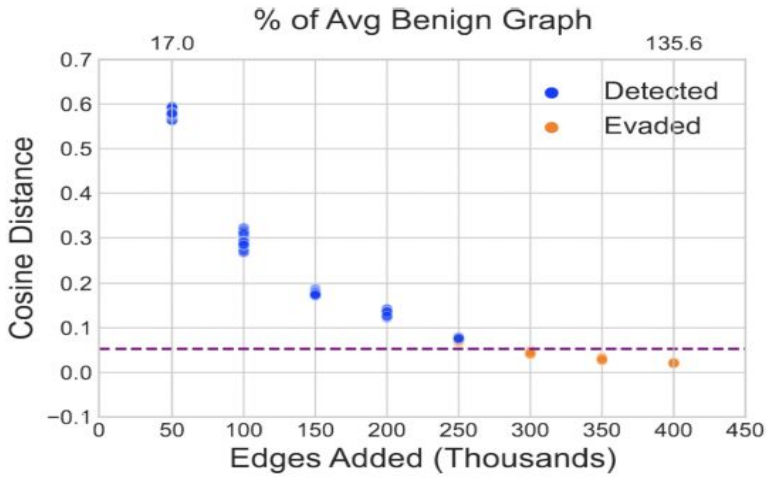
Gadget Implementation and Evasion Procedure Cont.

- Creating the evasion graph
  - Load the pre-attack graph
  - Load the benign graph
  - Find an insertion point
    - the set of edges within the attack graph that describes the attacker connecting to and gaining control over a process on the victim's machine
  - Inject benign substructures into attack graph
  - Insert the attack payloads substructures to the pre-attack graph

Evading StreamSpot

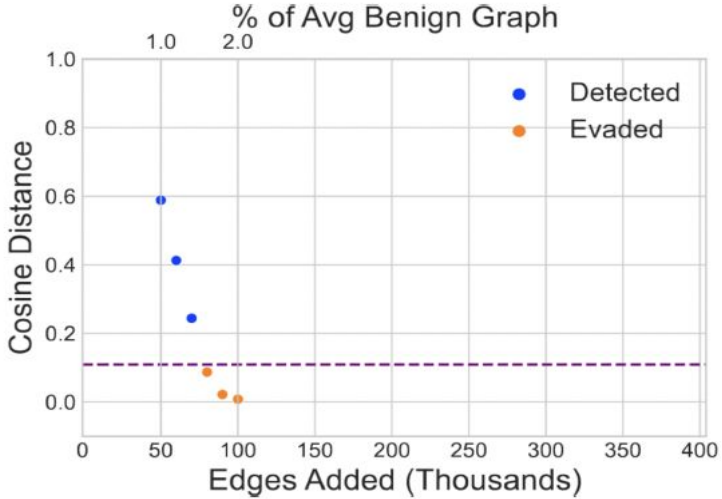
- Using the first gadget, abusing unweighted graph encoding
- Iteratively add batches of benign substructures to the attack graph until misclassification
- Continue to add more substructures to ensure that each attack remains undetected after the initial false negative.

StreamSpot



(a) StreamSpot Prov-HIDS

DARPA



(a) StreamSpot Prov-HIDS

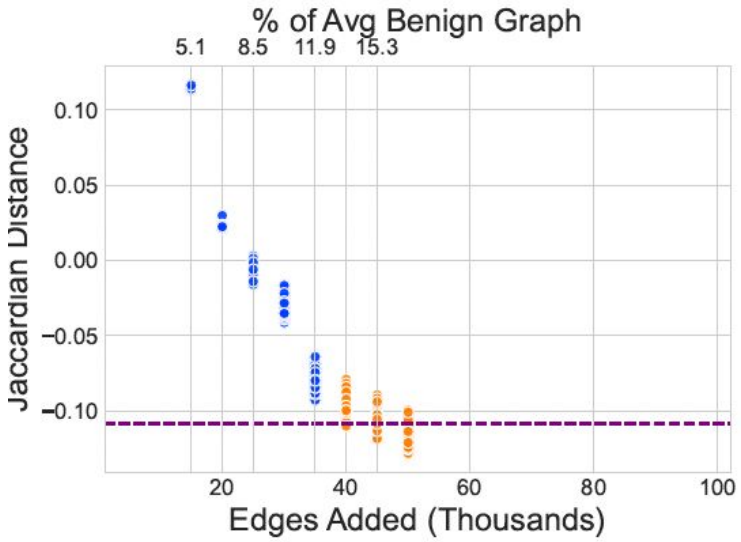
StreamSpot Remarks

- May seem unreasonable to add hundreds of thousands of edges to the attack graph
  - In reality, the attack scales to the size of the benign graph samples
- Difference in graph sizes may be the result of lack of diversity of benign activity in the former

Evading Unicorn

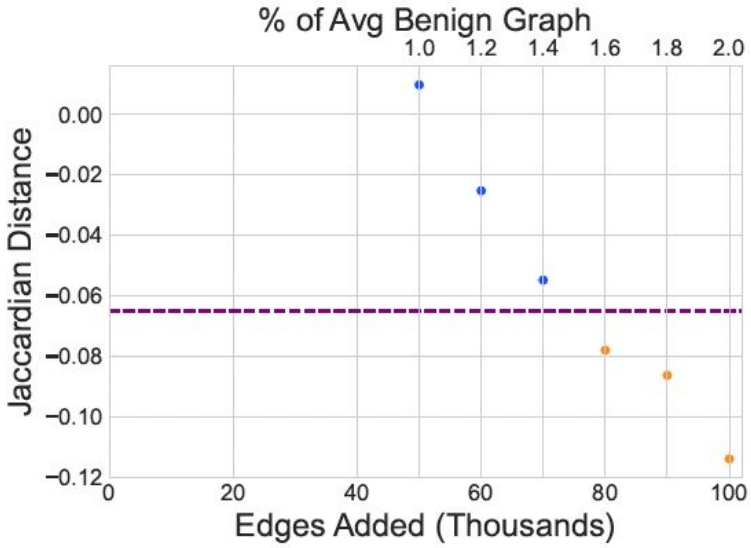
- Use the 2nd mimicry gadget, abusing distributional graph encoding
- Unicorn encodes the temporal properties of a provenance graph after t new edges
- For each batch of t edges in the graph that contains an attack edge, select t edges whose substructures match the distribution of the unmodified training graph

StreamSpot



(b) Unicorn Prov-HIDS

DARPA



(b) Unicorn Prov-HIDS

Evading ProvDetector

- Use the 3rd mimicry gadget, abusing downsampled graph encoding

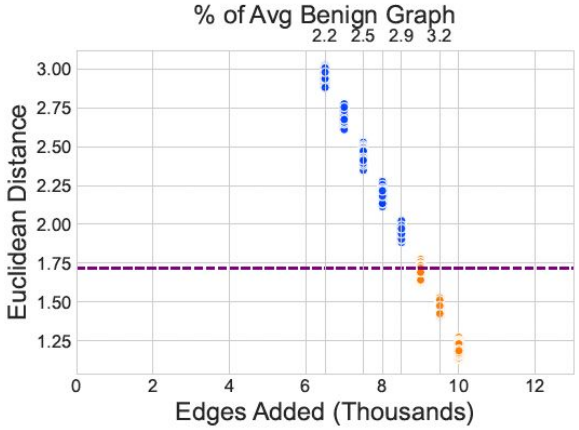
- ProvDetector classifies a graph as benign so long as more than K–N of its substructures fall within a known cluster
  - ProvDetector paper recommends K = 20 and N = 3

Evading ProvDetector Cont.

- Need to identified a benign path that exhibited lower regularity than all attack paths

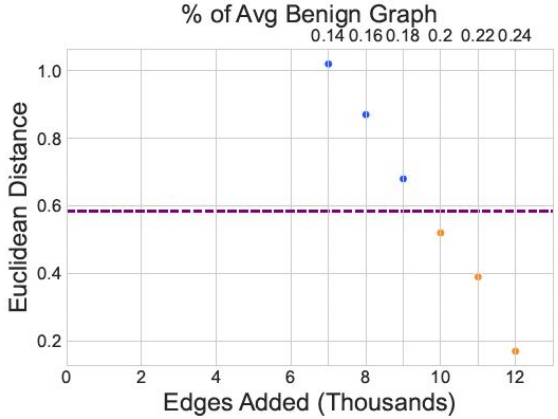
- Implementation repeatedly injects the substructures associated with this low-regularity benign path

StreamSpot



(c) ProvDetector Prov-HIDS

DARPA



(c) ProvDetector Prov-HIDS

Evading Pagoda

- Path based Prov-HIDS, it makes use of an event frequency database to assign a rarity score to individual edges
- If an edge in the path is not part of the frequency database, it is assigned 1, 0 otherwise
- A path's anomaly score is the average of all its edges' rarity scores
  - If anomaly score > threshold, flagged
- Also assigns graph anomaly score to entire graph, based on a weighted summation of all paths' anomaly scores.
  - If graph score > threshold, flagged

Evading Pagoda Cont.

- Use 1st and 3rd Mimicry gadget, abusing unweighted graph encoding and abusing distributional graph encoding

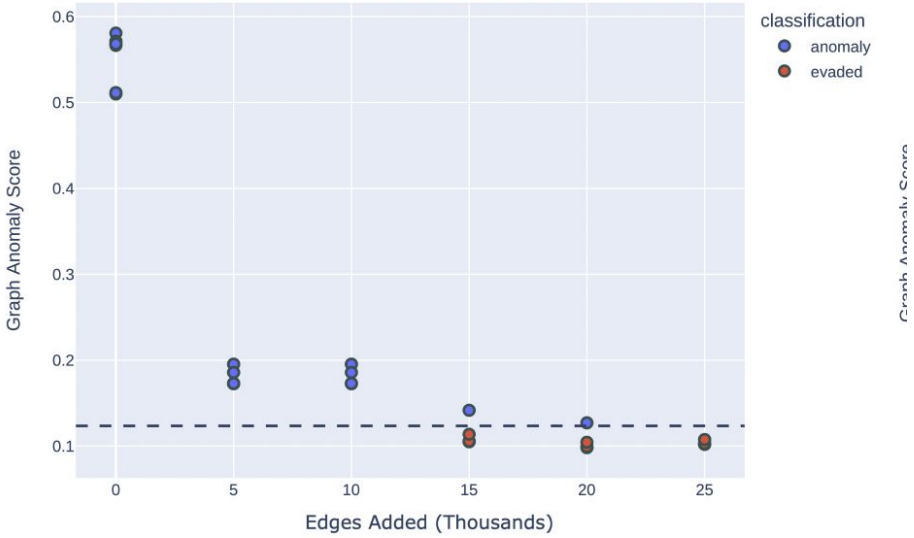
- An attacker can insert long benign paths to reduce the weights of shorter attack paths, thus lowering the graph anomaly score

Evading Pagoda Cont.

- First identify long benign paths

- Then, for each attack path, the attacker inserts edges from the frequency database to decrease the path anomaly score of the attack path

StreamSpot



(a) Pagoda

Full Graph Autoencoder (SIGL)

- SIGL is a Prov-HIDS that utilizes a graph autoencoder
- Graph autoencoders encode a vector and then attempt to reconstruct it
- A graph is labeled malicious if the distance from its embedding to the nearest training graph is greater than some predefined threshold

Full Graph Autoencoder (SIGL)

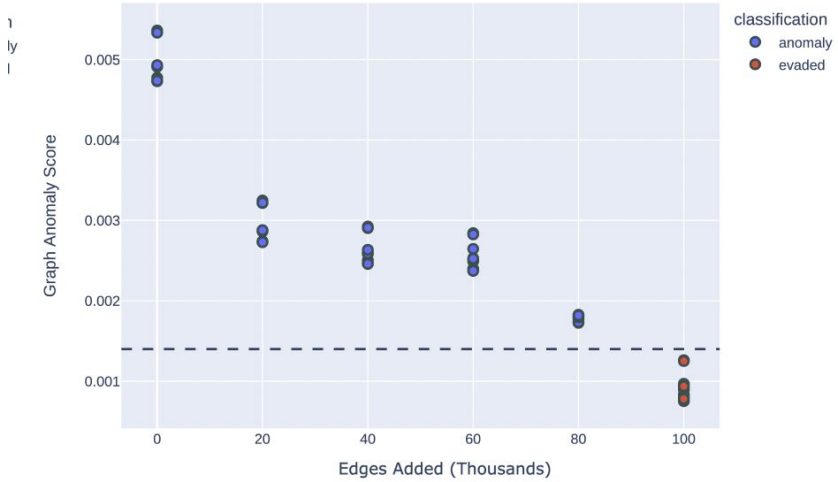
- On top of being close sourced, SIGLs use case is smaller and does not encompass a full graph.

- So the researchers implemented a Full Graph Autoencoder with the same ideas of SIGL for a more accurate comparison

Evading Full Graph Autoencoder

- Utilize the 2nd gadget, abusing distributional graph encoding
- Each node is encoded in terms of its “ancestral” k-hop neighborhood
- Adding the same “ancestral” neighborhood to an attack graph introduces nodes with benign embeddings
- This allows the average embedding of all the nodes in the attack graph to be closer to that of a benign graph

StreamSpot



(b) FGA

Combined StreamSpot, Unicorn, and ProvDetector

Evasion Strategy	E.A.	StreamSpot	Unicorn	ProvDetector
Gadget 1 (§V-B1)	$80K$	✔	✖	✖
Gadget 2 (§V-B2)	$80K$	✔	✔	✖
Gadget 3 (§V-B3)	$10K$	✖	✖	✔
Gadget 2 → Gadget 3	—	✖	✖	✔
Gadget 3 → Gadget 2	—	✔	✔	✔

TABLE III: Cross-comparison of evasion gadgets. ✔'s mark successful evasion, while ✖'s denote failure. E.A. refers to *Edges Added*.

Evasion Performance Under Incorrect Parameters

- Researchers considered an attacker with imperfect knowledge of the intrusion detection system
- Changed key parameters, StreamSpot's chunk size, Unicorn's sketch size, and ProvDetector's path length
- We selected these parameters, because they play a roughly analogous role in each system.

System	Parameter	Value	TPR	TNR	Evasion Rate
StreamSpot	Chunk Size	40	0.86	1.00	1.00
StreamSpot	Chunk Size	45	0.91	1.00	1.00
StreamSpot	Chunk Size	50	1.00	1.00	1.00
StreamSpot	Chunk Size	55	1.00	0.95	1.00
StreamSpot	Chunk Size	60	1.00	0.90	1.00
Unicorn	HistoSketch Size	500	0.99	0.94	1.00
Unicorn	HistoSketch Size	1000	1.00	0.92	1.00
Unicorn	HistoSketch Size	2000	1.00	0.93	1.00
Unicorn	HistoSketch Size	4000	0.95	0.96	1.00
Unicorn	HistoSketch Size	8000	0.29	0.96	1.00
ProvDetector	Path Length	4	0.0	1.00	1.00
ProvDetector	Path Length	6	0.21	1.00	1.00
ProvDetector	Path Length	8	1.00	1.00	1.00
ProvDetector	Path Length	10	1.00	1.00	1.00
ProvDetector	Path Length	20	1.00	0.92	0.86

Live Attack Demonstration

- Previous work has noted issues can occur when one attempts to invert samples from the feature space to the problem space.

- Replicate an attack scenario similar to the StreamSpot dataset
  - Benign activity was generated by opening the Firefox browser and visiting youtube and cnn

Live Attack Demonstration Cont.

- The victim clicks a malicious link that exploits a vulnerability in Firefox
- A reverse shell to open and creating an attacker-controlled process.
- Before initiating the remainder of the attack, the attacker attempts to evade the Prov-HIDS by injecting innocuous behavior patterns
- The attacker then injects its payload behavior

Methodology of Live Attack

- Two Attack types
  - a Full Knowledge (FK) attack and a Limited Knowledge (LK) attack.
- In FK attack the attacker can read directly from the system logs allowing them to identify the websites visited
- In the LK scenario, the attacker can infer that Firefox is running but cannot directly access the logs
  - The attacker browses the sites on their own browser to obtain and inspect similar output

Results

Evasion Strategy	StreamSpot	Unicorn	ProvDetector
Full Knowledge	✔	✔	✔
Limited Knowledge	✔	✔	○

TABLE V: Live demonstration results for limited knowledge and full knowledge attack scenarios: ✔'s mark successful evasion, and ○'s represent partial success (see details in §VI-K).

Note: ProvDetector is not triggered, but attacker path is in top 20 most anomalous paths.

Success because the most anomalous paths appear to be legitimate activities, therefore it is unlikely that the attack would be detected.

Comparison to Domain-General Attacks

- Results show that Prov-HIDS tested are vulnerable
- Not yet clear whether the specialized attack strategies are necessary
- Researchers then considered the applicability of domain-general graph evasion strategies

Attack methods and Results

Attack Method	Cls Tsk	Ptrb Typ	Mdl Acs	Grdnt Bsd?	Grph Typ	Prov-HIDS Compatible?
InfMax [56]	N	GA	C	No	Any	✖
EDA [57]	N	E	O	No	Any	✖
GF-Attack [58]	N	E	C	No	Any	✖
ReWatt [59]	N	E	C	No	Any	✖
EpoAtk [60]	N	E	O	Yes	Any	✖
AGA-GAN [61]	N	E	C	No	Any	✖
SRL [62]	N	E	C	No	Any	✖
CD-ATTACK [63]	N	E	C	No	Any	✖
Tang et al. [64]	G	GA	C	Yes	Any	✖
Xu et al. [65]	G	N+E	O	Yes	Any	✖
TNI [66]	G	N+E	O	No	Spl	✖
RL-S2V [24]	G	E	C	No	Any	?
Our Approach	G	N+E	O	No	Any	✔

TABLE VI: **Related Graph Evasion Attacks**— *Classification Task (Cls Tsk)* is either on nodes (N) or on the entire graph (G); *Perturbation Type (Ptrb Typ)* describes the graph properties transformed: Edge (E), Node (N), and Graph Attributes (GA); *Model Access (Mdl Acs)* denotes whether an open (O)- or closed (C)-knowledge model is assumed; *Gradient Based (Grdnt Bsd?)* indicates whether a graph neural network is assumed; and *Graph Type (Grph Typ)* designates whether a model can work on any (*Any*) graph type or only against a special (*Spl*) type.

Runtime Performance

Evasion Step	Time (sec)
Load the Pre-attack Graph	0.4334
Load Benign Graphs	0.0007
Find the Insertion Point	0.5490
Inject Benign Substructures	0.8325
Insert the Attack Payload's Substructures	0.0010
Total	**1.8166**

TABLE VIII: Runtime performance for each step in our evasion strategy.

This is only for StreamSpot Prov-HIDS on the StreamSpot dataset as it was used for comparison against a Domain-General method RL-S2V

Comparison to RL-S2V

m	Execution Time Budget	Past State Budget	Training Iterations	Time Per Iteration (sec)
1	7 days	50000	3,024,000	0.2
100	7 days	5000	201,600	3
1,000	7 days	500	20,160	30
10,000	7 days	100	2,240	270
100,000	7 days	10	604	1000

TABLE VII: Training performance of RL-S2V as the number of edge additions (m) increases. Each configuration was run for the same amount of time and the past state budget was modified to prevent it from hitting a memory wall.

Potential Mitigation Strategies

- Systems classifying lower-level graph structures such as nodes, edges, and subgraphs may be robust to the evasion strategies

- For whole-graph classification systems to become more resilient, security researchers may need to develop solutions that better leverage the properties of provenance graphs

Praise and Criticisms

Praise

- Open Source to serve as a baseline for future Prov-HIDS models
- Attacks are based on theory, not on specific code/implementation issues
- Provides sound reasoning on results and theory
- Experiments were thought out well and conducted thoroughly

Criticism

- Don’t go into reasoning as for why they chose the 5 specific “exemplar” systems outside the fact that they are unsupervised
- For closed sourced solutions, they had to recreate it from the research paper, which can pose issues that relate to human error.
- Not clear why they only tested Pagoda and SIGL against StreamSpot dataset
- Not enough datasets

Thank You for listening!

Questions?

References

Goyal, Akul, et al. “Sometimes, you aren’t what you do: Mimicry attacks against provenance graph host intrusion detection systems.” *Proceedings 2023 Network and Distributed System Security Symposium*, 2023, https://doi.org/10.14722/ndss.2023.24207.

$$f(G_i) = \mathbb{1}(\mathcal{F}^{\delta}(\mathcal{E}^{\kappa}_{\lambda}(\mathcal{N}^{\gamma}_{\beta}(G_i))) \geq \alpha)$$

- $N^{\gamma}_{\beta}(G_i)$ is a deconstruction function that's parameterized by γ and β that returns a set of substructures $z_i \in Z$ such that $z^j_i = N^{\gamma}_{\beta}(v^i_j)$ where $\forall v^i_j \in V_i$
    - γ is a branching factor that determines the size of v^i_j's neighborhood
    - β is a depth factor describing the max distance between a node in the substructure and v^i_j.
- $E^{\kappa}_{\lambda}(Z_i)$ is an encoding function that summarizes Z_i into an L-Dimensional vector V_i
    - κ specifies the size of a subset of Z_i used to represent G_i
    - λ is the embedding function used
- $F^{\delta}(V_i)$ is a distance function that compares V_i against a set of learned graph encodings $\delta = \{V^p_{(1)}, \cdots, V^p_{(n)}\}$, which returns the smallest distance between a graph in in δ and V_i
- α is a distance threshold, under which G_i is considered benign.