Visualization of Provenance System Log for Intrusion Detection

Junfeng Xu (jfxu@cs.ubc.ca), Michael Kim (megabyte@cs.ubc.ca)

November, 4, 2019 (revision v0.2)

1 Introduction

This project aims to visualize a provenance system log for a intrusion detection. There are numerous techniques to prevent malware, but recent development of malware makes the task more challenging. Especially, there are family of malware called advanced persistent threats (APTs) which attacks a target PC in slow and low fashion which could be difficult to be detected by system administrator in advance.

To solve this problem, there are researches on going to detect APT style program by using a system provenance log. Specifically, Camflow[1] which is maintained by researchers including UBC NSS lab could capture the system provenance log at system level. It hooks system calls using Linux Security Module (LSM) to describe the relationship between system nodes (file, sockets etc.).

It would be a promising research, however, there is a challenge that the researcher who wants to analyze the system provenance will face the high complexity of provenance semantics. We will tackle this challenge using data from UNICORN which is an unpublished advanced paper from Harvard. By visualizing the system provenance log in form of visualization, UNICORN framework will give us the information about the frequency of specific sub 'provenance graph', which information we could construct histogram from and map the each element of histogram to original system-wide provenance graph.

2 Visualization

Camflow, when set as recording whole system provenance data, produces around 0.5 MB/min of log data from target PC. Data follows the format from W3C Data model which is in JSON format.[2] It produces the relationship between system nodes in edge form. Thus one could imagine, millions of log in from of edges will come out as in the JSON format.

However, UNICORN framework abstracts the JSON format into a few encoded field as follow. It restricts the provenance information, which contains

the information from system time to operation type into the simple relationship encoding between nodes.

src ID	dest ID	src type	dest type	edge type
4	5	a	c	p
4	6	a	c	p
4	7	a	c	p
4	8	a	c	p
4	9	a	c	j

The framework calculates the frequency of similar sub-structure as follow. For a detailed explanation, we are doing back and forth communication with the PhD student from Harvard and our advisor Margo. Since a documentation for UNICORN is quite limited, we will keep update this document to improve that ambiguity.

hash value of sub-graph	frequency
177670	12
177671	75
177672	7065
177673	1

By analyzing the log, we want to investigate what vulnerabilities are exploited in the graph. If intrusion is detected at time period T, and provenance graph at T is G. At T-1 there is a similar graph G', we may want to find the difference between G and G'.

2.1 Domain Tasks

After talking to domain experts, we have identified the following domain specific tasks that can be performed using our system:

- Analyze the changes in histogram throught time, especially to identify the change in histogram that potentially leads to intrusion detections.
- Identify 'neighbourhoods' in the provenance graph.
- Map the histogram data back to the provenance graph

3 Implementation Plan

We plan to build a linked view that incorporates both the provenance graph, which is essentially a directed acyclic graph, and the generated histograms. Since the nodes are time-stamped, and that the histograms are generated over time, we will implement navigation through time so that the user can view the information at a earlier time.

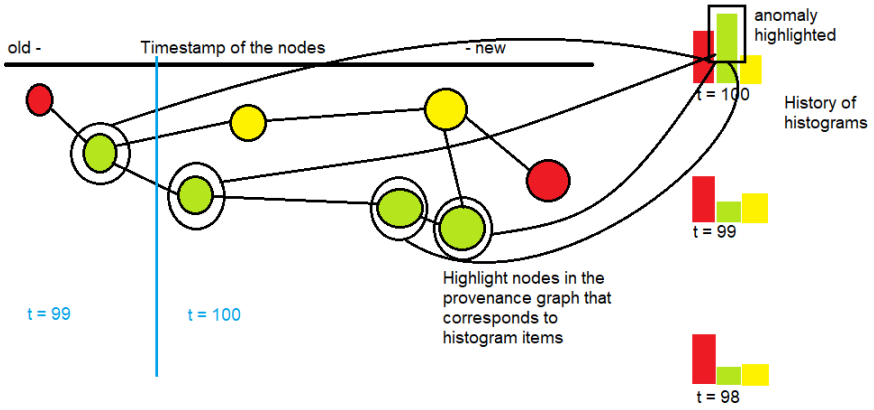

Figure 1: Rough sketch of the visualization interface, including the streaming provenance graph and the histograms.

We will investigate ways to highlight subset of nodes, and potentially the links and precursor to the nodes, as we build our implementation. Due to the large number of provenance graph nodes, we may also investigate how to reduce the number of nodes, by either aggregating nodes into close-knit 'neighbourhoods', only showing a slice of the nodes that are within a certain time interval, or filtering the nodes by some criteria.

3.1 Scenarios

After discussing the project with domain experts, we have established the following scenario in which our visualization can be utilized by an expert in the investigation of malicious program behaviour:

> After UNICORN detects an intrusion in a computer system, a security researcher is looking at the provenance graph log and histograms generated by UNICORN. The large number of data and seemly randomly changing histograms make it hard for the researcher to identify the location of the breach directly.
>
> By using our visualization, the researcher first looks at the histograms, where a spike in the number of provenance graph nodes with a certain label has been highlighted by the visualization. The researcher then looks back at histograms generated at earlier time stamps, and concluded that this anomaly is linked to the breach.
>
> Then, the researcher selects the label in the histogram and highlights all nodes in the provenance graph with the selected label. They then highlights the nodes added right before intrusion was detected. From the highlighted subset of nodes, the researcher was then able to identify the vulnerability in the system where the breach occurred.

3.2 Technologies and Libraries

We plan to build our visualization implementation as an interactive webpage. We plan to use the D3.js for managing the display of data, as one of us has previous experience with building diagrams using D3.js, and frontend development in general. To simplify the visualization workflow, We plan to perform all data processing in the browser using JavaScript.

3.3 Milestones

18 Nov establish the development environment, including version control, web development environment, and data processing setup.

18 Nov proof-of-concept node-link diagram visualization done

19 Nov peer review 1

25 Nov provenance graph visualization and highlighting done

26 Nov peer review 2

2 Dec linking between the provenance graph and histograms done

10 Dec project presentation

13 Dec project paper due

References

[1] Thomas Pasquier, Xueyuan Han, Thomas Moyer, Adam Bates, Olivier Hermant, David Eyers, Jean Bacon, and Margo Seltzer. Runtime analysis of whole-system provenance. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1601–1616. ACM, 2018.

[2] Khalid Belhajjame, Reza B'Far, James Cheney, Sam Coppens, Stephen Cresswell, Yolanda Gil, Paul Groth, Graham Klyne, Timothy Lebo, Jim McCusker, Simon Miles, James Myers, Satya Sahoo, and Curt Tilmes. Provdm: The prov data model. Technical report, 2012.