A First Look at Evasion against Provenance Graph-based Threat Detection

Zhenyuan Li [†], Runqing Yang [†], Qi Alfred Chen [¶], Yan Chen [‡]
[†] Zhejiang University, China [¶] University of California, Irvine, USA [‡] Northwestern University, USA

ABSTRACT

Provenance graph-based threat detection are widely studied as countermeasures against APT and other cyber threats for its powerful alert correlation capability. However, these detection approaches generally suffer from dependency explosion problem. And methods proposed to mitigate the problem pose underlying risks. In this poster, we first proposed a systemic mimicry attack approach against the underlying risks.

Then, we can generate adversary samples systematically. With these samples, we are able to provide a large dataset which can not only test the robustness of existing detection systems but also help with the design of new and more robust detection approaches.

1 INTRODUCTION

With the developing of advanced and stealthy attack technology, detecting attack at a single point gets much harder and more inaccurate. To overcome this challenge, security analyzers try to correlate the attack steps and detect the attack chain as a whole.

Provenance graphs are widely adopted for this purpose [1, 2, 5, 6] because of its ability to connect system-level nodes and behaviors. Provenance graphs represent the relationship between the control flow and data flow between the subjects (such as processes, threads, etc.) and the objects (such as files, registry, network sockets) in the system through a directed graph with timing.

1.1 Provenance graph-based threat detection and dependence explosion problem

Multiple approaches have been proposed for threat detection and analysis based on provenance graph, listed as following:

Backward tracking [4] and forward tracking are atomic operations on the provenance graph, with which researchers are able to locate all causally related nodes in the graph started from a suspicious node. However, with each additional step of tracking, the number of related nodes will increase exponentially. Thus, we always end up failing to track all nodes. This is the dependence explosion problem for provenance graph.

Tag-based alert correlation [2] stores local detection results in tags and correlates the attack chain through tag propagation in the provenance graph. Without control, tags will overspread and cause dependence explosion problem.

Graph alignment-based detection [6] is similar to the previous one, except that tags store local graph matching results, and therefore encounter the same problem. In this case, there could be multiple paths connecting the anomaly, resulting in a large number of false-positives.

Anomaly-based detection [1] differs from traditional anomaly detection in that it considers anomalies of the attack path rather than a single point.

1.2 Mitigation approaches for dependence explosion problem and underlying risks

To overcome the dependence explosion problem, researchers proposed several different mitigation approaches, listed as follows:

General data reduction [3] can mitigate the dependency explosion problem to some extend. However, general data reduction tend to keep the dependence between system objects to avoid compromising detection and thus have limited contributions to dependency explosion problem.

Decay-based pruning. [2, 6] For tag-based detection, the dependence explosion problem manifests itself in the overly broad passing of tags. Thus, researchers proposed decay-based pruning to limit the count of tag propagation rounds. Correspondingly, attackers can evade detection by lengthening the attack chain.

Anomaly-based path ranking. [1] Dependence explosion already brings on an overload of alerts. A straightforward idea to handle massive alerts is to rank them according to anomaly analysis and only deal with the top parts. However, multi-hop anomaly analysis recursively encounters dependence problem, and single-step anomaly analysis is easily bypassed.

Mitigation methods vary depending on the detection algorithms, but fall into the broad categories mentioned above and share the same limitations.

1.3 Mimic the whole attack chain against the underlying risks

Knowing the underlying risks, attackers can launch targeted mimicry attack. Main objectives include "live-off-the-land" that utilize more clean system tools and less dropped tools and extend the attack path as much as possible. Thereby, we assume that attackers can collect enough regular behaviors data to cover target mimicry attack behaviors.

In the following part of the poster, we propose a novel approach to systematically finding potential attack chains by representing attack stealthiness and feasibility as normalized scores between nodes and abstract the problem of finding attack chains into a problem of finding a path in an abstracted attack surface graph.

2 GENERATION OF MIMICRY ATTACK CHAINS AGAINST PG-BASED DETECTION

Researchers adopt various mitigation approaches to overcome the general dependency explosion problem of provenance graph-based threat detection. But the mitigation approaches introduce underlying risks. In this section, we propose a framework to investigate the underlying risks by mining potential stealthy attack chains (or attack surface) in system.

The attack chain mining process can be divided into three phases, discussed in the following subsections.

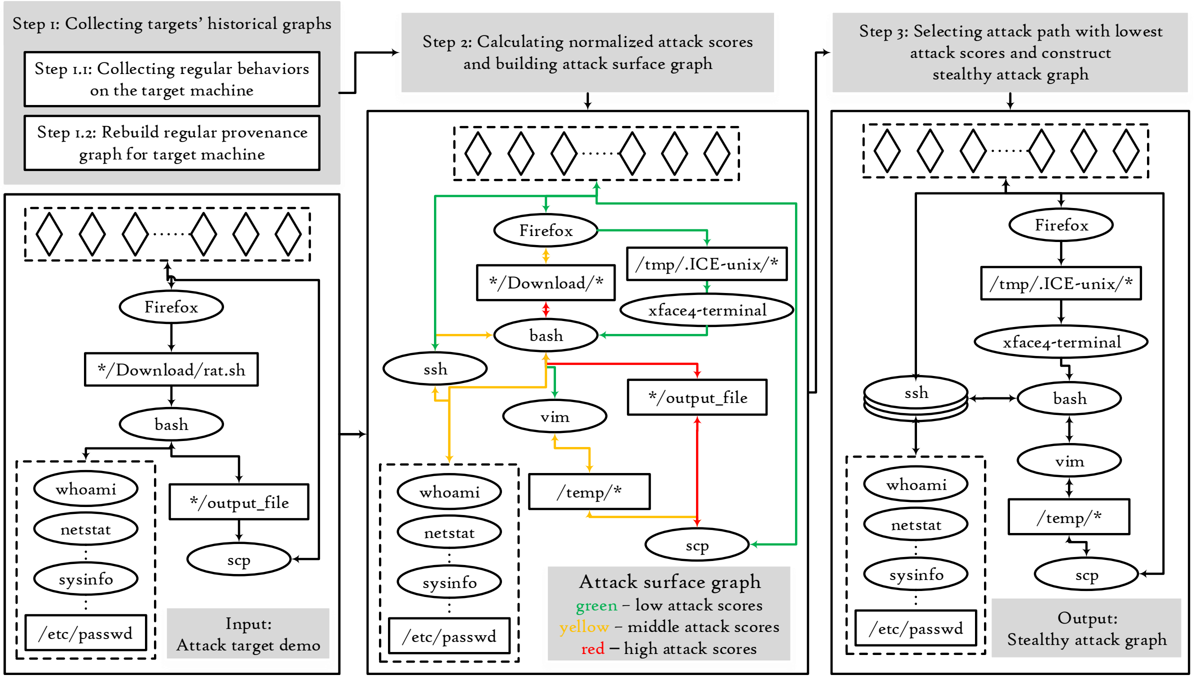
Step 1: Collecting targets' historical graphs
Step 1.1: Collecting regular behaviors on the target machine
Step 1.2: Rebuild regular provenance graph for target machine
Firefox
*/Download/rat.sh
bash
whoami
netstat
sysinfo
/etc/passwd
*/output_file
scp
Input: Attack target demo
Step 2: Calculating normalized attack scores and building attack surface graph
Firefox
/tmp/.ICE-unix/*
/Download/
xface4-terminal
bash
ssh
*/output_file
vim
whoami
/temp/*
netstat
scp
sysinfo
/etc/passwd
Attack surface graph
green – low attack scores
yellow – middle attack scores
red – high attack scores
Step 3: Selecting attack path with lowest attack scores and construct stealthy attack graph
Firefox
/tmp/.ICE-unix/*
xface4-terminal
ssh
bash
vim
whoami
/temp/*
netstat
scp
sysinfo
/etc/passwd
Output: Stealthy attack graph

Figure 1: Framework of our stealthy attack graph generation approach. (Diamond represent network objects, ellipses represent processes, rectangles represent file objects)

2.1 Collecting the target's historical graph

Based on the risks discussed in §1.2, intuitively, an attack that bypasses provenance graph-based detection should be close to regular behaviors. Thus, we first need to collect historical provenance graphs on the target machine. However, accessing the target machine's audit log is usually impossible.

An alternative approach is to collect information about the regular operations on the target machine and simulate them on a local machine. Then we can get a historical provenance graph that is similar to the target one.

2.2 Calculate normalized attack scores and build attack surface graph

We can encode several critical properties, including stealthiness, feasibility, etc., as distances between nodes in an attack surface graph. Then we can find ideal attack chains by finding the shortest path in the attack surface graph.

The distance in the attack surface graph can be calculated with a normalized attack score. As the following equation shows:

$$NS(s \rightarrow o) = log(\frac{Freq(s \rightarrow *)}{Freq(s \rightarrow o)}) - Decay + Feas(s)$$

Where $log(\frac{Freq(s \rightarrow *)}{Freq(s \rightarrow o)})$ represents the normalized proportion of events from subject s to object o in all events start from subject s. And $Decay$ and $Feas(s)$ denote the incentives for extending the attack chain and the difficulties posed for extending the attack chain, respectively. The last two items can be ignored in the preliminary experiments for convenience.

2.3 Searching attack chain in attack surface graph

To search for an attack chain, we need to identify the attack target at first. For example, as shown in Figure 1, the input target is to collect environment information with system command such as 'whoami', 'sysinfo', etc. As the pre-extracted attack surface graph shows, the original attack chain involves too many abnormal paths. And we can find much more stealthy attack chains as the output stealthy graph shows.

3 FUTURE WORK

We plan to conduct further research and experiments on the feasibility of implementing real-world attacks and the effectiveness of bypassing existing detection systems.

We hope that we can generate more adversarial attack samples and use them tho test the effectiveness and robustness of existing detection systems. Moreover, the generated samples can alleviate the problem of lack of samples from detection systems and thus help build better detection systems.

REFERENCES

[1] Hassan, W. U., and Guo, Shengjian, e. a. NODOZE: Combatting Threat Alert Fatigue with Automated Provenance Triage. In *NDSS' 19* (2019).
[2] Hossain, Md Nahid, e. a. Combating Dependence Explosion in Forensic Analysis Using Alternative Tag Propagation Semantics. Okland'20, pp. 1139–1155.
[3] Hossain, M. N., Wang, J., Sekar, R., and Stoller, S. D. Dependence-preserving data compaction for scalable forensic analysis. pp. 1723–1740.
[4] King, S. T., and Chen, P. M. Backtracking intrusions. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, ACM, pp. 223–236.
[5] Li, Z., and Chen, Qi Alfred, e. a. Threat Detection and Investigation with System-level Provenance Graphs: A Survey. *arXiv Computer Science* (2020).
[6] Milajerdi, Sadegh M., e. a. Poirot: Aligning attack behavior with kernel audit records for cyber threat hunting. In *CCS'19*.