Major improvements from the previous version

1. For this submission, we redo all the evaluations and add comparisons with other state-of-the-art causality analysis techniques (e.g., CPR [25], ReadOnly [33], Prio-Tracker [13], and NoDoze [18]) to demonstrate the effectiveness of DepImpact in revealing attack sequences.
2. In our evaluations, we collect the system audit logs from 2 hosts, and DepImpact achieves effective graph reduction results. The evaluation results demonstrate that the performance of DepImpact is not affected by the number of deployed hosts. We also provide a detailed discussion about how DepImpact can support causality analysis across multiple hosts or systems in Section VI-D.
3. We clarify the scope of our work in the threat model in Section III. We assume that the kernel and the kernel-auditing layer are trustable, and kernel hardening techniques [38,39] can be used to better protect the kernel.
4. We provide a detailed discussion on the possible evasion attacks in Section VI.
5. We add evaluations on how the top-ranked entry nodes help prevent the loss of critical edges by increasing the number of chosen entry nodes for forward causality analysis. The detailed results and analysis are in Section V-C.
6. To demonstrate the effectiveness of the features, we add a comparison evaluation between DepImpact and its simplified versions with fewer features in Section V-D. DepImpact achieves significant improvement over the simplified versions. Our evaluation in V-B also shows the superiority of DepImpact over other state-of-the-art causality analysis techniques.
7. We add the system performance comparison between DepImpact and NoDoze in Section V-E and discuss the possible way to further improve the performance of DepImpact in Section VI-C.
8. To better present our work, we formally define the technical terms (critical edge, attack entry, critical component, dependency weight, dependency impact) used in DepImpact.

Reviews from USENIX Security 2020

Review #158A

Review recommendation

2. Reject and resubmit

Writing quality

3. Adequate

Reviewer interest

2. I might go to a talk about this

Reviewer expertise

1. No familiarity

Paper summary

The paper proposes a system, DEPPROP, to trim large causality analysis graphs down to a few critical edges. The approach leverages three edge features (data size relevance, temporal relevance, and concentration ratio), which the paper argues are correlated with the POI. It then uses k-means clustering on these features to split edges into two groups: one likely containing critical edges, the other one containing non-critical edges. LDA is used to compute edge weights using the three features. Then for identifying critical edges, the system uses weight propagation, ranks the nodes, and then carries out forward causality analysis on the high-ranked nodes. The subgraph obtained identifies the critical edges. The authors evaluate the approach on 10 attack scenarios, and show that DEPPROP achieves >90% reduction rate, while only missing a few critical edges.

Strengths

On the evaluated set of 10 attack scenarios, the system achieves significant graph reduction, while correctly identifying the critical edges.

Weaknesses

The premise of the paper is that the critical edges have correlated features with the POI, which can be leveraged to trim the causality graph significantly. The paper picks three such features (data size relevance, temporal relevance, and concentration ratio), and shows that these work well for the given attack scenarios. The paper does not provide any evidence about the generality and robustness of these features (or ideas about additional features that could work).

Instead it argues, that: "However, it is important to note that the goal of this work is not to design highly effective and robust features to detect attack activities, which is an orthogonal and challenging problem in the presence of adversarial knowledge and adversarial dynamics."

I am not sure the system is useful if it based on features that work only in a handful of attack scenarios.

Evaluation: I found the evaluation to be insufficient and missing certain details:

The evaluation is done on logs from only one system. Real-world forensic investigations would likely involve analysis across several systems. The authors should either justify why evaluation on one system is sufficient or add more servers / scenarios to the evaluation set-up.

For the real-world users using the system (that provide non-critical edges), you list 100 million events but it is unclear how many users and processes these are from? Is it representative of real-world logs?

How many critical edges are there in each of the ground truth scenarios?

    - The evaluation does not provide any discussion on in what scenarios and why DEPPROP would miss certain critical edges? How could the system be improved to account for these missing critical edges?

Detailed comments for authors

In general, the paper is well-written and easy to follow. I outline the main weaknesses of the work in the sections above. Here I provide some additional comments / points that need clarification:

- Section 5.1.1:
    - The "Commonly used exploits" that you list are attack scenarios rather than exploits.
    - The following are unclear:
    - What are the tools used in illegal storage vs. illegal storage (2)?
    - What does "unsafe" server mean here?
- Section 5.1.2: Real attacks: The term "lateral movement" is used incorrectly. Lateral movement means the attacker attempts to move to other devices in the network, but there is only one server involved in the scenarios you list.
- Section 5.2:
    - Evaluation Results: It is unclear what are the POIs for each of the listed attacks.
    - Is Figure 3 for data across all the attacks?
    - Similarly, in "clustering results of edges" it is unclear which attack's reduced dependency graph is the section talking about, when it says the graph has more than 40K edges.
- The system uses existing graph reduction techniques as a first step, but it is unclear how much reduction is already offered by this step. Please add these numbers.
- Writing: The paper uses at times: "empirically, we use a threshold of x" (for example, 10 seconds) without providing the supporting empirical evidence.
- Writing Nit: a lot of use of "as such"

Requested changes

- Please list additional features that the system can leverage, or argue that the proposed features are sufficient for a wide range of attack scenarios
- Please justify the evaluation set-up and add the missing details listed in the weaknesses section above

Review #158B

Review recommendation

3. Major revision

Writing quality

3. Adequate

Reviewer interest

2. I might go to a talk about this

Reviewer expertise

2. Some familiarity

Paper summary

This paper presents DEPPROP, a system to reduce the size of dependency graphs for causality analysis on user space audit logs. The system identifies critical components as attack sequences and removes non-critical edges based on their edge weights (with respect to a certain threshold). To evaluate DEPPROP, the authors deployed the system on a server and performed a series of exploits that inject malicious payloads through key system interfaces and three real attack cases and provide evidence that DEPPROP is an efficient tool to remove irrelevant edges.

Strengths

this is an important problem for forensic investigations and system event reconstruction
the approach is well described; figures and graphs are useful

Weaknesses

the paper could benefit from removing redundant information.
some system design decisions are not sufficiently justified
important details on the evaluation are missing (ground truth, more information on the single server that the approach was evaluated on)

Detailed comments for authors

First and foremost, this is an interesting research problem with practical relevance. Causality analysis for attack detection is widely used and resource intense. Thus, reducing the graph size by filtering irrelevant edges is an important problem. Related work has considered different approaches to filter out well known dependencies between processes but some of them have only been proven to work under certain experimental conditions. Therefore, I consider this incremental work a useful contribution to the body of literature.

I think that the paper is generally well-written but it is very repetitive. I strongly recommend the authors to reduce redundant information as far as possible. It is not a problem if the paper is shorter especially if this makes it easier to read.

Section 4.2.3 on edge weight computation describes the steps in a detailed way and explicitly defines certain parameters. The choice of these parameters however is not sufficiently justified, e.g., why did the authors chose k=2 and n=20?

I appreciate the choice for exploits used in the evaluation, but unfortunately it is difficult to understand how well the results generalize as the evaluation is based on one server only. Also, the server log data is not sufficiently described in order to understand whether this experimental setup was actually ecologically valid. Especially when comparing your approach to related work it is necessary how well the framework performs on different servers and log files. I also think that one server only is not sufficiently to draw conclusions that generalize to different types of servers in large real-world networks.

It is also not clear what the authors consider as ground truth and how many "false positives" are considered as irrelevant edges.

Discussion: Please extend the discussion with respect to related work. It is rather obvious for me that the graph size could be even further reduced by combining this approach with other works. It remains to show whether related work filters out a different subset of irrelevant edges (e.g., by considerring re-occuring patterns). if this is the case, a combination of multiple approaches could be a major improvement to the state of the art.

I appreciate that the authors define a threat model which is in line with related work. However, I strongly encourage the authors to discuss the impact of an untrusted kernel on the overall results to outline the frontiers of this approach.

Review #158C

Review recommendation

3. Major revision

Writing quality

3. Adequate

Reviewer interest

2. I might go to a talk about this

Reviewer expertise

3. Knowledgeable

Paper summary

In this paper, the authors describe a system for working backwards from POIs (points of interest i.e. artifacts) discovered by other systems/incident responders/etc. The system is intended to help discover root cause of the creation of a POI or event.

Strengths

Solution is highly relevant to incident response techniques

Features used are intuitive and may be similar to existing manual processes

Weaknesses

Poor system performance analysis

Evaluation is unclear and does not prove that the system can scale

Detailed comments for authors

The paper describes DepProp, a system for discovering critical paths in event dependency graphs. These types of analyses are particularly useful for identifying root cause for incidents.

The authors evaluate their system with by performing attacks described in other literature, then process the dependency graphs with DepProp.

The evaluation contains some clarity issues, which raise questions about the scalability of this system. For example, when the authors execute the attacks, what is the timeframe between the beginning of the logs (used for the dependency graph) and the end of the logs? Figure 1 seems to imply that the logs begin at the "root cause" and end at the POI, though I suspect this is not actually the case. Does the evaluation benefit from only needing to examine a small timeframe of events, where the beginning and end are both known? The authors need to clearly describe their methodology in Section 5.

Furthermore, the system performance seems reasonable (5.3) -- I would not complain about waiting 430s on average to save potentially hours of time. Average time measurements do not provide enough information to know how this time scales (particularly with regard to the problem listed above). If a week or a month passes between an initial infection vector and a POI, does it take hours/days to process the graph? The paper indicates that these tasks could be done in parallel, but the complexity of the entire system should be described -- in order to both understand its utility and for future comparisons with this work.

The work has some clear limitations. The authors discuss that attackers could deliberately adjust the "size" of individual events to evade data size evaluation, for example. I am curious if the authors have any ideas for countermeasures against this type of behavior and if these could lead to future work. In Section 4.2.1, the authors discuss how edges are merged; is this also a limitation (i.e., could an attacker mimic other read/write I/O sizing to cause attack edges to be merged with benign edges)?

The authors claim that one advantage of this system is that the system reduces the scope of investigation from hundreds of thousands of edges to a few (Section 1). I suspect this is overclaimed. I am unconvinced that real-world security analysts examine hundreds of thousands of possible events to discover root cause. Instead, I suspect they use expertise and heuristics to narrowly search for related events. Claims of saving time or effort in real world investigations need to be substantiated (data, user study, etc.).

Review #158D

Review recommendation

2. Reject and resubmit

Writing quality

3. Adequate

Reviewer interest

2. I might go to a talk about this

Reviewer expertise

2. Some familiarity

Paper summary

The authors tackle a problem in computer forensics - that of helping investigators work backwards from a point of interest (e.g., the discovery of a malware file in someone's account) to determine how all of the steps in the attack. They key to their approach is their ability to work backwards through the graph, building it up to all possible entry points, rank those entry points, and then work forward again in order to prune the graph to a small enough number of nodes and edges that a forensic investigator is able to examine the results.

Strengths

Significant reduction in the (graph) data that an investigator needs to examine
Demonstrates approach on a live system (e.g., multiple users) with ten known attacks

Weaknesses

No discussion of the possible impact of the "false negatives" (parts of the graph that were missing from the final set of critical nodes and edges)
Requires additional discussion on the (user) selection of the number of entry points
Lack of a user study (or even case study) with forensic investigators

Detailed comments for authors

The authors describe an interesting problem in computer forensics - how do you reduce the set of events on a system to those that are related to a specific point of interest? The approach taken is interesting a demonstrates a significant reduction in the amount of data a forensic scientist would need to examine.

I am generally concerned, however, with how easy this would be to deploy and use. For example, what are the different variables that users can adjust and what are the effects from that? In section 4.2.3, step 1, you mention that you chose n=20 due to emperical analysis. Is this generalizable? Or will this kind of analysis need to be performed for each new data set to be examined? Similarly you provide results when the top 1, 2 and 3 points of entry are chosen, but also note that this results in some nodes and edges being missed (see Table 5). How might a user determine how many entry nodes they should select? What are the sizes of the resulting graphs when all of the entry nodes (for a given attack) are chosen? What is the lowest ranking for an entry node that is part of an attack? I would like to see more discussion and analysis regarding the choice of entry nodes.

In addition to discussing the choice of entry nodes, the authors need more discussion regarding the missed nodes and edges. Given that the ground truth is known, can the authors comment on the impact of the missed entry points when, for example, only the top ranked entry point is chosen for an attack? How much of an impact might that have on a forensic analysis? What are the trade-offs between having more (irrelevant) data to investigate versus missing parts of the attack?

This paper would be improved if the authors were able to perform at least a case study with a forensic analyst who can provide usability feedback on the approach.

On page 10, under the comparison approach with the monkey approach, you state that "For the monkey approach, all the entry nodes are candidates ...." Did you randomly select from all of the entry nodes, or only from the top nine? If the former, I'm surprised by how good the results are, especially for the number of missing nodes and edges. Can you please either clarify and, if random from all the entry points, discuss why the results are still so good?

The comparisons with existing approaches felt somewhat ad hoc. The closest full comparison was with PrioTracker in Section 5.2.4, but this comparison was short (e.g. Table 8 only provided the fanout weight distribution) when compared to, for example, the monkey approach (Table 7). I would like to see more discussion on how your approach is better than other approaches (along with any limitations, such as missing entry points).

I was a bit confused by Figure 3 on page 9. It looks like 100,000 edges have a weight = 0, which represents approximately 55% of the edges. But then there are another 100,000 edges with a weight = 0.9, but this is only about 20% of the edges? Am I misinterpreting what the percentage is representing?

Nits:

On page 9, "scrap files into a single compress file and transfer it" should be "scrapes files into a single compressed file, and transfers it...."

On page 11, "analysi for reduction" should be "analysis for reduction"

On page 12, "to remain stealth" should be "to remain stealthy"

Requested changes

1. Greater discussion around the impacts of the user selected values (e.g., number of entry points), and the trade-offs between including irrelevant entry points versus missing entry points.
2. Ideally a case study with a forensic investigator who uses your tool.

Review #158E

Review recommendation

2. Reject and resubmit

Writing quality

4. Well-written

Reviewer interest

1. I am not interested in this paper

Reviewer expertise

2. Some familiarity

Paper summary

This paper aims as countering cyber-attacks by deploying a form of intensive monitoring networks and using causality analysis to detect vulnerabilities. To this end, a novel framework called DEPPROP is developed that identifies the critical component of a dependency graph.

Strengths

- the paper is well written
- real systems data was collected and used
- performance results are good

Weaknesses

- the lack of more concrete explanation
- the results are hard to evaluate

Detailed comments for authors

This work is well written and seems to be relevant for monitoring industrial control, cloud (and other) systems. While the authors do a good job in outlining the framework on some general and conceptual level, I miss some more concrete explanation on its deployment. For example, for a specific system how does this process of assigning weights to dependencies in order to distinguishing relevant dependencies work? It is not an easy task for a reader to understand it sufficiently in order to implement it.

The authors mention a total of 20k lines of code, which suggests a substantial effort in making it. Would those 20k lines of code be made available at some point? There is no way to verify this and the authors only mention building the framework “upon Sysdig [53]”, which is a reference to a website that is even not a proper link.

There seems to be some related works but there is no meaningful comparison with those in terms of effectiveness, performance etc.

I believe that clarifying on those issues would improve this work substantially and could lead to acceptance.