Appendix
Example of the Ground Truth Labeling

In this supplementary material, we demonstrate the advantages of our labeling process with a concrete example from Day 2 of the red team campaign.

The first step of our algorithm is to manually identify a set of input events in raw logs, based on the description in the ground truth document. We will use the following entry for this example:

> *09/24/19 15:04:14 – On Sysclient0005 via RDP session, exported 3.5 gb exfil file allgona.zip*

This journal entry contains two indicators of compromise that we use in our search: the name of the target host `SysClient0005`, and the file name (which contains a typo) of the exfiltrated data dump `allgone.zip`. Searching for all operations involving this file, we discover a process with PID 6864 and command line string "`movingonup.exe fun.com 81`", which we use to form the input set S_p for our algorithm.

In the theoretical model described in section III-B, child-parent process relations are *weak* entity dependencies that require manual analysis. The parent of the initial process is `cmd.exe`, which we mark as malicious, and continue to its parent, which is `Explorer.EXE`.

The red team document states that the activity on host `SysClient0005` is performed through an RDP connection, which means that the execution of this process is just part of the standard interactive user session, and it is not malicious *by itself*. The benign label of the `Explorer.EXE` process is written into the output set of labels to make sure that the labeling algorithm can be deterministically replayed. The discovered process tree can be visualized as Fig. 6(a), with the initial process indicated in red, and the benign process in green.

However, the red team could have executed other malicious processes during the same RDP session, so all children of the `Explorer.EXE` process are added to the queue for manual analysis, and in the next loop iterations, the security analyst identifies several other benign processes associated with background and autorun tasks, consistent with the baseline of the environment.

But some of the child processes contain clear indications of the malicious activity, and are labeled as such by the analyst. The process tree discovered so far is shown in Fig. 6(b).

The last step of our algorithm is to unconditionally propagate the malicious label to all child processes of a labeled malicious process, which is done without any input from the analyst. Fig. 6(c) shows the resulting process tree with **24 malicious** and **14 benign** labeled processes.

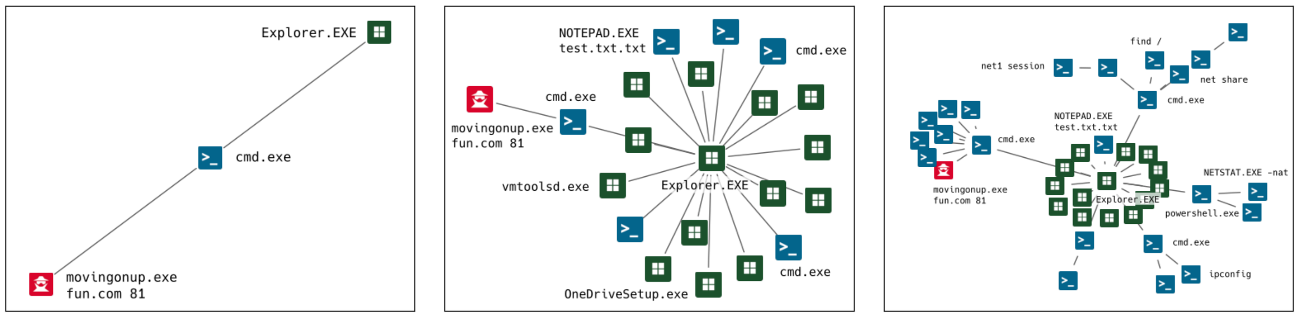
The algorithm continues to scan logs from other hosts to discover any additional processes that communicate with the same malicious IP addresses, and the origin of the RDP session. We do not show these further iterations here due to the space constraints.

Inspecting the labeled malicious processes reveals instances of the reconnaissance activity shown in Table I. These actions of the red team operator **are not listed in the ground truth document**, highlighting the problem of the incomplete ground truth, as discussed in section II-A. But because all these processes are linked through a *strong* dependency, we can confidently rule out the possibility of normal operations performed by a system administrator in the same time frame.

To complete the labeling process, we propagate the malicious label to all raw events associated with the discovered processes. This gives us **3,468** events in total, which have all been deterministically derived from a **single input event**, and associated with a specific line in the ground truth document.

TABLE I
INSTANCES OF UNDOCUMENTED RECONNAISSANCE ACTIVITY

Host	PID	Process
SysClient0005	7384	`net share`
SysClient0005	7988	`find /`
SysClient0005	6688	`NETSTAT.exe -nat`
SysClient0005	7912	`net1 session`
SysClient0005	7512	`ipconfig`
SysClient0005	6040	`NOTEPAD.EXE test.txt.txt`



(a) (b) (c)

Fig. 6. Iterative discovery of malicious processes, starting from a single entry in the red team journal. The red element indicates the initial input of the search algorithm. Green elements are processes marked as *benign* by an analyst, including the `Exlorer.EXE` process at the center of the graph. The remaining is the malicious processes executed by the red team.