CS261 Project Proposal and Research Plan

Kenny Yu & R.J. Aquino

10/06/13

Proposal

Given the extensive amount of data and history that PASS and other existing provenance solutions collect, we believe that an intrusion detection system built with PASS data could be as effective, if not better, at detecting intrusions than existing intrusion detection systems (such as Somayaji's system to monitor system call sequences). We believe that provenance storage systems *should* have some form of intrusion detection built into the system, because systems that require provenance data typically place a high concern on the integrity and security of the data and would want to know if data have been modified through an intrusion.

The research questions we are investigating relate to the provenance collected specifically by PASS. More explicitly:

- Is the data collected by PASS sufficient for detecting intrusions?
- If yes, what kinds of intrusions can PASS detect?
- Can PASS detect more types of intrusions than Somayaji's system?
- Would an intrusion detection system built with PASS data be more accurate in detecting intrusions than existing intrusion detection systems?
- What provenance data, in particular, aids in detecting intrusions?

We believe these questions are interesting to explore because this work will allow provenance storage and collection systems to kill two birds with one stone:

- provide a way to track the history of files
- provide a way to detect if the data have been tampered with through an intrusion

We also believe it is exciting to explore the potential of all the data collected by systems like PASS, and to explore the kinds of applications that can be built with this provenance data.

These questions are important because the answers to these questions will allow us to determine the types of intrusions that can be detected, and exactly what kinds of provenance data would be needed to detect these intrusions. This will allow engineers to design and build

more robust provenance storage and collection systems with intrusion detection, and possibly intrusion response, automatically built into the system.

These questions qualify as research because there are currently many different types of provenance storage and collection systems and many different types of intrusion detection systems, but it is not clear which of these models is the right model, if at all, for combining provenance with intrusion detection. We hope to investigate PASS data to see the limits of the types of intrusions that PASS can detect, and then provide some generalization about provenance requirements for intrusion detection.

Related Work

Background work

For background work, we focused on provenance and intrusion detection. We focus on the paper that introduces PASS, and Somayaji's work on using system call sequences to detect intrusion. Below we list the major takeaways from the relevant background work, as it relates to our plans.

Reddy: Provenance Aware Storage Systems (2006)

- http://www.eecs.harvard.edu/syrah/pass/pubs/usenix06.pdf
- Original paper introducing PASS
    - collects all data about the environment of a program, as well as process-process and process-file dependencies when creating/modifying a file
- Paper talks about PASS v1, we both have played around with PASS v2 for A1.

Somayaji: Automated Response Using System-Call Delays (2000)

- http://www.cs.unm.edu/~immsec/publications/uss-2000.pdf
- This is the original paper describing his method of:
    - monitoring sequences of system calls from all processes
    - use this to build a "normal" profile of a process
        - a profile consists of a bit vector, where a bit position represents a pair of system calls that appear together
    - when a process deviates from this profile, exponential slow-down the running process
    - their profiles can adapt when the pattern of system calls start changing (but not an intrusion)
- Problems with his approach:
    - potentially takes a very long time to build the normal profile of a process
    - profiles are stored on disk, and are not checked if they are tampered with
    - when processes implement user-level threads, the system calls of all the user-level threads seem interleaved and messy to the kernel, and it is harder and

takes longer to build a normal profile
    - only way of defending against an attack is slowing down; no support for different kinds of actions to take or different types of attacks

Somayaji: Lookahead Pairs and Full Sequences: A Tale of Two Anomaly Detection Methods (2007)

- http://people.scs.carleton.ca/~soma/pubs/inoue-albany2007.pdf
- This is his more recent work on intrusion detection
- compares lookahead sequences with full sequence analysis
    - also introduces random schema masks: “mask” out certain syscalls in a window. (random sampling)
- good grab bag of techniques for analyzing system call sequences

Contextual work

For contextual work, we looked for papers that combined both provenance and intrusion detection. Our goal was to see the different kinds of ways existing systems attempt to detect intrusion, and what kinds of provenance data they need, if they use provenance data. In order to keep our list organized and succinct, we present it in a bulleted list, highlighting the relevant takeaways from the papers (techniques that we can use for our approaches, and problems with their techniques).

Cao: Intelligent Fuzzy Anomaly Detection of Malicious Software (2012)

- http://www.engr.uky.edu/~mqiu/PI_papers/published_papers/9.%20IJAI_published.pdf
- Problem this paper tries to solve: hard to define boundaries between normal and abnormal behavior
- Solution: fuzzy anomaly detection - instead of looking for a clear boundary (true or false), data is mapped to a range. Doesn't tie itself to a specific technique, but can be applied to a number of techniques. The papers in this paper's citation list deal more specifically with techniques.
    - If you have a technique already, here’s how you can modify it and (possibly) make it better

Tariq: Identifying the Provenance of Correlated Anomalies (2011)

- http://web.engr.illinois.edu/~tahir2/publications/identifying-provenance.pdf
- Problem: How can we trace anomalies back to individual files and processes that caused the anomaly, and how can we do it efficiently?
- They present the idea of an “anomaly-provenance bridge” to map anomalies back to their provenance in a distributed system
    - space efficient representation for mapping tuples by noticing that the same provenance record (e.g. open file) occurs many times
    - hash table where keys are anomalous k-tuples (a fingerprint), and values are a linked list of nodes

    - a node contains a pointer to a provenance record in a database
    - this data structure might be useful for us***
- During a training period, they record k-tuples for their system to "characterize normal behavior". During detection, a k-tuple that was not recorded is "anomalous"
- Focused more on managing this process in a distributed system, but the Related Works section highlights background work on detecting normal vs. abnormal behavior in a system (including, e.g. a database of previous system call patterns, similar in spirit to Somayaji).

Reddy: Provenance as first class cloud data (2010)

- http://dl.acm.org/citation.cfm?id=1713258
- presents motivation for tracking provenance data in a cloud
    - also briefly mentions intrusion detection systems

Oliner: Community Epidemic Detection using Time-Correlated Anomalies (2010)

- http://adam.oliner.net/files/oliner_raid_2010.pdf
- Problem: how can you detect an epidemic in a community (multiple instances of the same application)
- Their solution: use statistical properties of a community (homogenous set of instances of an application) for time-correlated anomalies
    - each client has a function, anomaly signal
    - they report their anomaly scores regularly to a monitor
    - monitor aggregates all the client's scores and analyzes the distribution of scores
    - outliers/statistical properties indicate anomalies/epidemics within the community

King: Enriching Intrusion Alerts through Multihost Causality (2005)

- http://www.eecs.umich.edu/techreports/cse/2004/CSE-TR-496-04.pdf
- Problem: how can we enrich IDS with source of attacks and potentially impacted hosts?
- Their solution:
    - build causal graphs for intrusion-related events in a distributed system
    - backwards: detect the cause
    - forwards: detect who is/will be affected
    - within a host, they build a dependency graph of files, processes, and filenames
        - exactly like PASS! but PASS keeps more detailed information
        - we can use their technique to identify the source of intrusions and potential impacted files
        - their limitations: they don't log everything, and this leads to covert channels. PASS will fix this!

Conclusion Format

First, we will seek to claim that PASS data can correctly identify exploits. So, our conclusion might read "An analysis of PASS data correctly identified 95% of exploits run, and incorrectly identified 15% of normal activities as possible exploits." Secondly, we will want to

compare our result to that of Somayaji, so we will have to state that "Somayaji identified 80% of exploits that PASS correctly identified", or similar. Finally, we intend to generalize about which forms of provenance data are useful for detecting intrusions. Towards that end, we will make conclusions like "using X and Y, we detected 75% of intrusions, but using X/Y/Z, we detected 70%. Z is not useful for detecting intrusions." By running our experiments with enough subsets of the provenance data, we can hopefully make strong claims about the usefulness of certain forms.

Experimental Setup

To answer the questions: "Is the data collected by PASS sufficient for detecting intrusions?" and "If yes, what kinds of intrusions can PASS detect?", we will run a small set of exploits and then analyze the data in the provenance through ideas from our background and contextual work, and some of our brainstorming. These ideas include (but are not limited to):

- analyzing system call sequences (like Somayaji)
- analyzing file access patterns
- analyzing file-process and process-process dependencies
- using machine-learning techniques (e.g. to determine what "type" of file we are working with, or what "type" of process we are running)
- a combination of the above

By running these experiments, we hope to learn whether it is feasible to use PASS data to detect intrusions, and determine an automatic way of doing so with high accuracy.

To answer the question: "Would an intrusion detection system built with PASS data be more accurate in detecting intrusions than existing intrusion detection systems?", we will run experiments of this form:

1. Run exploits on our provenance-collecting system. The exploits will be a combination of attacks taken from the Somayaji paper and the other works listed above, as well as attacks run through Metasploit, an open-source community managed exploit system.
2. Analyze the provenance data aggregated during the exploits and determine if our intrusion detection system can correctly identify whether an intrusion occurred.
3. Run the exploits on other intrusion detection systems (e.g. Somayaji), and determine if these other systems can correctly identify whether an intrusion occurred

Our goal with this experiment is to compare accuracy of our provenance-based IDS with existing intrusion detection systems.

We intend to compare our system to Somayaji's. The pH (process homeostasis) kernel modifications he describes in his paper are available online, so we hope to run it and compare our results to his. Namely, we intend to measure the true/false positive/negative rates for the exploits we run on our PASS system, as a basis of comparison. In order to do this, we will need to apply Somayaji's kernel patch to a VM running Linux (ideally Ubuntu 9.04 if it has the right kernel version). We will then run exploits and use the user-level tools Somayaji provides to determine which processes his system flags as malicious.

We will begin by running a small set of exploits to see if we can detect exploits that our provenance-aware intrusion detection system has not already seed. As described in Cao above, there are two main classes of exploit detection - misuse detection (for known exploits) and anomaly detection (for unknown exploits). We will need to closely examine the provenance generated from a representative sample to determine which data might be useful for our analysis. We will ultimately test each element of the provenance data that we think might be useful, to determine whether it actually is useful.

Conditional on whether we can detect intrusions using PASS data, we will answer the question: "What provenance data in particular aids in detecting intrusions?" as follows. We hope to identify the provenance record types that we believe will aid in intrusion detection. After we identify the relevant record types, we will measure and report what percentage of our provenance data is of each type. We will then run our intrusion detection on the generated data multiple times, each time having our intrusion detection system take into account different subsets of the data (broken down by provenance record type, as described above) and measuring the false/true positive/negative rate for each subset. This technique will allow us to determine which types of provenance data are most useful for detecting intrusions, and which forms of data are too noisy or inconsistent to be useful. The goal of this experiment would be to determine which provenance data should be included in a new provenance-based intrusion detection system, based on success rates and the size of the provenance data.

Conditional on whether we can detect intrusions using PASS data, we will answer the question: “Can PASS detect more types of intrusions than Somayaji’s system?”. We will attempt to run workloads and exploits that Somayaji’s paper describes as difficult for his pH system, in particular user-level threads (the kernel interprets the intertwined system calls in user-level as one big mess of system calls). At first, we will test our system as it was designed for basic exploits. If, like Somayaji, we find that workloads containing user-level threads elude our detection system, we will attempt to fine-tune our approach to specifically search for the provenance data that a user-level threaded exploit generates. Our goal with this experiment is to learn about the types of intrusions exist, and how our system and Somayaji’s system can or cannot detect these.

In order to fully evaluate our system, we will run increasingly complex exploits. Ideally, our system will detect with 100% accuracy the simple exploits it was trained on. We will increase complexity/variation, running both "known" and "unknown" exploits to generate PASS data. We will then run our detection system on the generated PASS data, and compare our accuracy (false/true positives and false/true negatives) with other intrusion detection systems.

We will run these tests in the PASS Ubuntu 9.04 VM. This affords us a few benefits - first, it is already fitted with PASS. Second, it is an old/unsupported version of Ubuntu, so there are a number of exploits we can run that have not been patched, allowing us more diversity with respect to our experiments.

Through the above experiments, we will be able to answer the questions laid out in our proposal. Specifically, we will be able to determine whether PASS data is sufficient to identify intrusions, and to what extent it is able to reliably do so. To measure reliability, we will run an exploit concurrently with different amounts of “benign” processes and determine the accuracy of our intrusion detection system based on the provenance collected from this combined workload.

Across all the experiments, there are a few problems we hope to plan for and avoid. First, we may run into trouble finding and running exploits on our system. Secondly, and more importantly, the provenance data generated by the exploits may be so large that it is difficult to determine a good direction for using the provenance data to identify our exploits. With regard to this, we are considering falling back on an unsupervised machine learning approach, if we can not identify useful data by hand to run a supervised test. Finally, at a high level, we should be concerned about how secure our systems and techniques are. Can they be fooled? Because PASS is built into the filesystem, our data shouldn't be tainted, but in theory a crafty attacker with knowledge of PASS and our techniques could manipulate the PASS data to cover their tracks.

Resources Needed

We will need relatively little in the way of resources for this project. We intend to use the prepackaged PASS VM (running on Ubuntu 9.04), available here: http://www.eecs.harvard.edu/syrah/pass/download/v2/ . This will allow us to collect the provenance data needed to run our experiments. We have already set up these VMs and set up the necessary mirrors, so that we can install external software, even though it is an unsupported version of Ubuntu.

We will also need to install Metasploit (http://www.metasploit.com/), to run exploits on our virtual machine. In early testing, we were able to get Metasploit installed, and are working out a few bugs to make sure it will run.

Finally, as we want to compare our work to Somayaji's, we will have to set up his system. His source code is available as a diff of the Linux kernel here: http://people.scs.carleton.ca/~mvvelzen/pH/pH.html.

Schedule

Below we list tentative due dates for various components of our project:

Sunday, 10/6 - Proposal Due
Wednesday, 10/9 - finish all background/contextual readings
Saturday, 10/12 - get PASS setup to run exploits

Saturday, 10/12 - identify/acquire code for exploits (other than Metasploit)
Sunday, 10/13 - Run exploits on PASS, start to analyze provenance data

Week of 10/14 - First project meeting, revise remainder of schedule
Tuesday, 10/15 - identify potential techniques for determining "good" or "bad"ness of a process from its provenance data, and begin to try them out.

Monday, 10/21 - Have a rudimentary system in place to analyze large sets of provenance data
Tuesday, 10/22 - In class project report, Quiz #2

Monday, 10/28 - Determine specific process for determining whether a process is "good" or "bad". Depending on our results from previous weeks, this may take the form of a machine learning solution, or a more specific approach.
Friday, 11/1 - Finalize overall process.

Week of 11/4 - Second project meeting
Monday, 11/4 - Have "switch"able system in place, so that we consider only certain types of provenance, regardless of system determined above
Friday, 11/8 - Start to run system on a subset of our exploits, Set up Somayaji's system

Tuesday, 11/12 - Determine whether system is working sufficiently
Friday, 11/15 - Monday 11/18 - Run final experiments

Tuesday, 11/19, 5pm - First draft due
Thursday, 11/21 - In-class presentations

Tuesday, 11/26 - Quiz #3, check in with course staff with regard to what we should do in the final weeks of the project

Tuesday 12/10 - Final project due