Using Provenance for Security and Interpretability

Xueyuan Han
Harvard University
Cambridge, Massachusetts
hanx@g.harvard.edu

ABSTRACT

System security is somewhat stymied because it is difficult, if not impossible, to design system defenses that address the full complexity of a system's interaction. Interestingly, this problem has parallels in understanding how machine learning (ML) algorithms make predictions. Both of these problems require a structured, comprehensive understanding of what a system/model is doing. My dissertation addresses these seemingly disparate problems by exploiting data provenance, which provides just such a solution. I exploit provenance both to design intrusion detection systems and to explain how ML algorithms arrive at their predictions.

1 INTRODUCTION

Security systems such as intrusion detection systems (IDSes) often require an understanding of system execution, using observed information flows to make critical security decisions. While the techniques for modeling system execution to obtain such an understanding become increasingly sophisticated [5], the underlying information used remains mostly unchanged. For example, a significant number of host-based IDSes use system call traces to model system behavior, evolving from simple enumerations of system call sequences [8] to variable-length n-gram modeling [15], and from first-order Markov models [37] to their high-order counterparts [16]. As security applications continue to fail to deliver stronger security guarantees (e.g., in terms of detection and false positive rates), researchers recognize the need to improve the quality of the underlying information and design systems that analyze the context of system execution, in addition to the execution itself. For example, tools that monitor system calls take into consideration arguments passed into a system call [22] and its caller function [36]. However, such additions do not change the nature of the information; they still provides the security system with only a single layer of semantics and a linear description of system activities. Furthermore, interactions within and between processes are difficult to untangle, which results in an ambiguous and sometimes incomplete picture of system execution. As such, desirable data for security analysis such as intrusion detection must present a complete, structured view of system execution that is amendable to incorporate different layers of semantics as such a need arises. It must describe a detailed history of normal activities for the system to decide whether a future event is an attack or not.

Similarly, the use of history to make future decisions is also applicable to the field of machine learning (ML). Complex ML models achieve high accuracy at the cost of *interpretability* [19], thus offering little transparency to justify predictions and to detect unwanted unfairness and discrimination [7]. Conceptually, an ML model is constructed to understand from the training data the intricate association between observations and conclusions. Once such an association is established, the model can be regarded as a complicated "system" and any labelled data validated by the model reveals the history of a correct "execution" of the "system". Such a history of "execution" provides a means to explain why a future prediction is made. Unlike current approaches that focus on explaining only local behavior of the model [17, 29], this approach to providing explanations presents a global perspective that allows for a better interpretation of how the model behaves under various scenarios, as long as the model is thoroughly validated.

In both cases, *provenance* is the ideal source of data. Provenance was originally used to describe the origination and chain of ownerships of works of art [28]. Computer scientists have adapted the term to refer to special metadata describing how digital objects came to be in their current state. As methods of capturing digital provenance have evolved [25, 28], provenance has been suggested as a source of information for various applications ranging from explaining the existence of data in database systems [12], to detecting and potentially preventing breaches in computer systems [13]. I intend for my dissertation to advance the state of the art in leveraging data provenance as the foundation for system security and model interpretability. In computer security, the fact that provenance offers a complete, causal history of the execution of a system [11] makes it potentially a rich source of information for detecting system intrusions and identifying or constraining the flow of sensitive information. In ML, provenance is the history of model validation. It provides clues for inferring inner workings of ML models that associate observations with conclusions, thus enabling interpretation that assists fair and justified decision-making [21].

The remainder of the paper is organized as follows. In section 2, I discuss how data provenance enhances system security, while in section 3, I discuss its ability to enable ML interpretability. I present my current research agenda in section 4 and conclude in section 5.

2 SYSTEMS SECURITY

Modern computer systems deploy a variety of security tools to create multiple layers of defense, including a *prevention* layer (e.g., firewalls, intrusion prevention systems), a *detection* layer (e.g., IDSes), and a *reaction* layer (e.g., anti-virus software) [26]. Security tools in all these layers use detailed audit trails provided by the operating systems and applications to enforce security policies and detect intrusions. Such information is usually the foundation of security applications, the quality of which determines the efficacy of security measures. Unfortunately, state-of-the-art security tools fail to guarantee integrity, confidentiality, and availability of the

EuroDW'18, April 2018, Porto, Portugal
2018. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

system, even though the algorithms they deploy become increasingly sophisticated [1, 23, 30]. My hypothesis is that the underlying audit information that those tools depend upon has become one major bottleneck in further improving their efficacy and data provenance provides a new solution to the problem. As such, I would like to demonstrate the advantages of data provenance for a range of security techniques.

Audit data viewed at different layers of abstraction leads to different semantic views of actions. For example, when a user escalates privileges by executing the program `su`, the application level log usually presents a single-line high-level view of the action:

```
Jan 31 07:36:53 Michaels-MBP.local su[91231]: SU Michael to root on /dev/ttys000
```

The system call level log, however, generates a long list of entries that detail various actions taken by the operating system, including obtaining the user's effective `UID`, and opening and checking the password file [4].

Although useful on their own merits to tackle a subset of security problems that are visible to their level of abstraction [24], audit trails fail to provide security systems with layered semantics that are required to understand from top-down (or bottom-up) the complete picture of system activities. Such a holistic understanding not only improves detection accuracy and precision, but also allows for attack attribution, enabling system administrators to reason over the detected intrusions. For example, an IDS that analyzes function call level workflows [10] can identify stealthy aberrant path attacks that alter the normal execution path of a function [32]. In many cases, these attacks are constructed as data-oriented attacks [14] that do not change the control flow of the entire program, thus making it difficult for system call based IDSes to detect. On the other hand, monitoring only function calls cannot defend against system call injection or return-oriented attacks that are discernible by those IDSes. However, naively combining two IDSes so as to use information from both layers does not provide a better solution because

i) a single intrusion could cause redundant alarms that overwhelm the human operator;

ii) it is challenging to correlate system call level alarms with the function call level workflow, which is more interpretable for the operator;

iii) fundamentally, audit data from either level is unstructured, therefore unable to present interactions between processes.

Provenance, however, creates a *structured* view of execution at various semantic levels since it can represent the causal relationship of system execution as a directed acyclic graph (DAG). It can also be layered to incorporate different levels of abstraction while ensuring overall consistency and connectivity [10, 24]. By reasoning and correlating information at various layers, security tools can be made *accurate* in detecting various intrusions, *resilient* to common evasion attacks (e.g., mimicry attacks [27, 35]), and *interpretable* for attack attribution.

In the rest of the section, I briefly describe two different security tools that can be improved through provenance and the challenges associated with the approaches.

IDS. Continue from the previous IDS discussion, a provenance-based IDS can simultaneously analyze function call and system call traces, taking into account their correlations as represented in the DAG. The DAG also elucidates the interactions between various processes, untangling the intricate dependencies among them.

Challenges. The size of provenance data can grow very large very quickly, especially in a distributed setting. The computation of provenance data carried out by the IDS must be efficient enough to detect the intrusion before it wreaks havoc on the system. My previous work [13] attempted a window-based approach to limit the amount of data to be analyzed. However, it has been shown that this approach could lead to information loss [2]. Provenance data is also an ideal source of forensic evidence. Efficiently storing and querying provenance for post-mortem analysis is an important step for provenance-based security analysis [38].

For concurrent systems where nondeterminism frequently occurs, it is difficult to reason provenance exactly. It is possible to address this challenge through *fuzzy* approaches and statistical measures [34].

Honeypot. A honeypot is a trap that masquerades as a production system to lure attackers to probe, exploit, and compromise while collecting valuable information of their activities [33]. A useful honeypot therefore must be able to record detailed traces of those activities to help security experts understand how the attack took place. State-of-the-art honeypots use virtual machine introspection (VMI) [9] to stealthily capture the trace of an attacker [20], but suffer from the semantic gap problem as the virtual machine monitor has only a low-level external view of the monitored system [6]. A provenance-based honeypot offers a comprehensive picture of the attack that is manifested by not only high-level steps but also actions taken "under the hood."

Challenges. Capturing provenance in the monitored system requires us to deploy sensors either as user processes or in the kernel [9]. Compared to the VMI approach, it is relatively easy for the attacker to detect, manipulate, and disable provenance capture [31]. In addition, any provenance data stored in the system can be tampered with. The former challenge requires at minimum a stealthy, efficient deployment of a provenance capture mechanism, while the latter requires a combination of cryptographic techniques and a covert, just-in-time delivery method.

3 MACHINE LEARNING INTERPRETABILITY

Provenance is an ideal solution to system security because it describes a complete, causal history of what happened on the system, which can be used as a reference to *predict* what is expected to happen in the future on the same system. We can also take advantage of this concept of using history as a basis to predict expected future behavior to explain ML models, because any prediction must be derived from the model's understanding of historical data during training. The question then is how we can reveal such an understanding from the model, which consists of merely mathematical equations. The key insight is that model validation reveals the model's inner workings and the history of how a data is validated shines a light on its understanding of historical data.

The training process allows the model to learn the associations between the observations and conclusions from the training data and encode them in mathematical formulas. Once the model is fitted (i.e., the model "obtains" an understanding), we can consider it as a "system", and any labelled data validated by it is a correct

"system execution" that reflects its understanding. Provenance in this context describes the history of model validation for every labelled data (i.e., how an output value derives from an input data through the model), providing a rich corpus from which we can reason about a future prediction. When a prediction is made, we can use its provenance to identify similar "executions" that support this prediction and use those labelled data to explain why the model behaves in a certain way.

4 CURRENT RESEARCH DIRECTION

To date, I have designed a behavior-based intrusion detection system that analyzes at runtime system-level provenance graphs to detect application anomalies [13]. I focused on Platform-as-a-Service (PaaS) users who run many instances of an application on many compute nodes to collect reference provenance graphs for anomaly detection. By comparing subgraphs using a modified label propagation algorithm, I generated a model of the application. Any instance that did not conform to the model was identified as an anomaly. Although the experimental results have shown to be promising, I have encountered several limitations that are inherent to this approach. For example, the window-based approach to analyze subgraphs inevitably broke the connectivity of the graph, disregarding long-span program behavior [32]. Since only low-level provenance data was monitored, it was difficult to attribute the high-level cause of the intrusion.

To overcome the limitations of the previous approach and to advance the state of the art in intrusion detection with data provenance, I am currently designing an IDS with the following goals:
i) Correlating provenance at different layers of abstraction to detect at runtime various attacks that can evade detection at one level but not the other (or the combination thereof).
ii) Provenance generates abundant data, which is ideal for machine learning to generalize normal behavior. A suitable machine learning algorithm designed with provenance domain knowledge can minimize false positive rates while maintaining high detection accuracy.
iii) Layered provenance capture enables interpretable attack attribution at a high level, even when an intrusion alert stems from low-level provenance analysis. The system administrators can reason about the cause of an intrusion to reject false positive alarms, effectively making the IDS more usable [18].

5 CONCLUSION

I discussed how provenance is the ideal source of data to enhance system security and enable machine learning interpretability. Prior research has successfully designed many systems to capture provenance at different degrees of granularity [3, 10, 25]. The next step is to adapt provenance capture, storage, query, and analysis to apply it in the field of computer security and machine learning, as provenance itself renders little value without such applications. I plan to explore these possibilities for the duration of my Ph.D., and welcome suggestions and inspirations from experts in those fields and in computer systems in general.

REFERENCES

[1] Abdulla Amin Aburomman and Mamun Bin Ibne Reaz. 2017. *A survey of intrusion detection systems based on ensemble and hybrid classifiers.* Vol. 65. Elsevier Limited. 135–152 pages.

[2] Leman Akoglu, Hanghang Tong, and Danai Koutra. 2015. *Graph based anomaly detection and description: A survey.* Vol. 29. Springer Netherlands. 626–688 pages.

[3] Adam M Bates, Dave Tian, Kevin RB Butler, and Thomas Moyer. 2015. Trustworthy Whole-System Provenance for the Linux Kernel.. In *USENIX Security Symposium.* 319–334.

[4] Matt Bishop. 2003. *Computer security: art and science.* Addison-Wesley Professional.

[5] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM computing surveys (CSUR)* 41, 3 (2009), 15.

[6] Brendan Dolan-Gavitt, Tim Leek, Michael Zhivich, Jonathon Giffin, and Wenke Lee. 2011. Virtuoso: Narrowing the semantic gap in virtual machine introspection. In *Symposium on Security and Privacy (SP).* IEEE, 297–312.

[7] Lilian Edwards and Michael Veale. 2018. Enslaving the Algorithm: From a "Right to an Explanation" to a "Right to Better Decisions". (2018).

[8] Stephanie Forrest, Steven A Hofmeyr, Anil Somayaji, and Thomas A Longstaff. 1996. A sense of self for unix processes. In *Symposium on Security and Privacy.* IEEE, 120–128.

[9] Tal Garfinkel, Mendel Rosenblum, et al. 2003. A Virtual Machine Introspection Based Architecture for Intrusion Detection.. In *Internet Society Network and Distributed Systems Security Symposium*, Vol. 3. 191–206.

[10] Ashish Gehani and Dawood Tariq. 2012. SPADE: support for provenance auditing in distributed environments. In *Proceedings of the 13th International Middleware Conference.* Springer-Verlag, Inc., 101–120.

[11] Laurent Georget, Mathieu Jaume, Frédéric Tronel, Guillaume Piolle, and Valérie Viet Triem Tong. 2017. Verifying the reliability of operating system-level information flow control systems in linux. In *2017 IEEE/ACM 5th International FME Workshop on Formal Methods in Software Engineering (FormaliSE).* IEEE, 10–16.

[12] Muhammad Ali Gulzar, Matteo Interlandi, Xueyuan Han, Mingda Li, Tyson Condie, and Miryung Kim. 2017. Automated debugging in data-intensive scalable computing. In *Proceedings of the 2017 Symposium on Cloud Computing.* ACM, 520–534.

[13] Xueyuan Han, Thomas Pasquier, Tanvi Ranjan, Mark Goldstein, and Margo Seltzer. 2017. FRAPpuccino: Fault-detection through Runtime Analysis of Provenance. In *Workshop on Hot Topics in Cloud Computing (HotCloud'17).* USENIX.

[14] Hong Hu, Shweta Shinde, Sendroiu Adrian, Zheng Leong Chua, Prateek Saxena, and Zhenkai Liang. 2016. Data-oriented programming: On the expressiveness of non-control data attacks. In *Symposium on Security and Privacy (SP).* IEEE, 969–986.

[15] Neminath Hubballi, Santosh Biswas, and Sukumar Nandi. 2011. Sequencegram: n-gram modeling of system calls for program based anomaly detection. In *Third International Conference on Communication Systems and Networks (COMSNETS).* IEEE, 1–10.

[16] Wen-Hua Ju and Yehuda Vardi. 2001. *A hybrid high-order Markov chain model for computer intrusion detection.* Vol. 10. Taylor & Francis. 277–295 pages.

[17] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. *arXiv preprint arXiv:1703.04730* (2017).

[18] Christopher Kruegel, Darren Mutz, William Robertson, and Fredrik Valeur. 2003. Bayesian event classification for intrusion detection. In *19th Annual Computer Security Applications Conference.* IEEE, 14–23.

[19] Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. *arXiv preprint arXiv:1606.04155* (2016).

[20] Tamas K Lengyel, Justin Neumann, Steve Maresca, Bryan D Payne, and Aggelos Kiayias. 2012. Virtual Machine Introspection in a Hybrid Honeypot Architecture.. In *Workshop on Cyber Security Experimentation and Test (CSET).* USENIX.

[21] Zachary C Lipton. 2016. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490* (2016).

[22] Federico Maggi, Matteo Matteucci, and Stefano Zanero. 2010. Detecting intrusions through system call sequence and argument analysis. *IEEE Transactions on Dependable and Secure Computing* 7, 4 (2010), 381–395.

[23] Preeti Mishra, Emmanuel S Pilli, Vijay Varadharajan, and Udaya Tupakula. 2017. *Intrusion detection techniques in cloud environment: A survey.* Vol. 77. Elsevier. 18–47 pages.

[24] Kiran-Kumar Muniswamy-Reddy, Uri Braun, David A Holland, Peter Macko, Diana L MacLean, Daniel W Margo, Margo I Seltzer, and Robin Smogor. 2009. Layering in Provenance Systems.. In *USENIX Annual technical conference.*

[25] Kiran-Kumar Muniswamy-Reddy, David A Holland, Uri Braun, and Margo I Seltzer. 2006. Provenance-aware storage systems.. In *USENIX Annual Technical Conference.* 43–56.

[26] Marcin Nawrocki, Matthias Wählisch, Thomas C Schmidt, Christian Keil, and Jochen Schönfelder. 2016. A survey on honeypot software and data analysis. *arXiv preprint arXiv:1608.06249* (2016).

[27] Chetan Parampalli, R Sekar, and Rob Johnson. 2008. A practical mimicry attack against powerful system-call monitors. In *Proceedings of the symposium on Information, computer and communications security.* ACM, 156–167.

[28] Thomas Pasquier, Xueyuan Han, Mark Goldstein, Thomas Moyer, David Eyers, Margo Seltzer, and Jean Bacon. 2017. Practical whole-system provenance capture. In *Proceedings of the 2017 Symposium on Cloud Computing.* ACM, 405–418.

[29] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1135–1144.

[30] Atmaja Sahasrabuddhe, Sonali Naikade, Akshaya Ramaswamy, Burhan Sadliwala, and Pravin Futane. 2017. Survey on Intrusion Detection System using Data Mining Techniques. (2017).

[31] Stewart Sentanoe, Benjamin Taubmann, and Hans P Reiser. 2017. Virtual Machine Introspection Based SSH Honeypot. In *Proceedings of the 4th Workshop on Security in Highly Connected IT Systems*. ACM, 13–18.

[32] Xiaokui Shu, Danfeng Daphne Yao, Naren Ramakrishnan, and Trent Jaeger. 2017. Long-span program behavior modeling and attack detection. *ACM Transactions on Privacy and Security (TOPS)* 20, 4 (2017), 12.

[33] Lance Spitzner. 2003. *Honeypots: tracking hackers*. Vol. 1. Addison-Wesley Reading.

[34] Lefteri H Tsoukalas and Robert E Uhrig. 1996. *Fuzzy and neural approaches in engineering*. John Wiley & Sons, Inc.

[35] David Wagner and Paolo Soto. 2002. Mimicry attacks on host-based intrusion detection systems. In *Proceedings of the 9th Conference on Computer and Communications Security*. ACM, 255–264.

[36] Kui Xu, Ke Tian, Danfeng Yao, and Barbara G Ryder. 2016. A sharper sense of self: Probabilistic reasoning of program behaviors for anomaly detection with context sensitivity. In *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 467–478.

[37] Stefano Zanero. 2004. Behavioral intrusion detection. In *International Symposium on Computer and Information Sciences*. Springer, 657–666.

[38] Wenchao Zhou, Micah Sherr, Tao Tao, Xiaozhou Li, Boon Thau Loo, and Yun Mao. 2010. Efficient querying and maintenance of network provenance at internet-scale. In *Proceedings of the 2010 SIGMOD International Conference on Management of Data*. ACM, 615–626.