Research Statement

Wenchao Zhou
http://www.cs.georgetown.edu/~wzhou/

Introduction

The past decades have witnessed the pervasive adoption of computer systems that have greatly influenced almost every aspect of human society. The increasing popularity comes with a massive demand for computation and storage resources that far surpasses the capability of a single computer. In reality, these computer systems are often deployed in a distributed manner, consisting of a collection of computers that are distributed across multiple administrative and geographical domains. This makes designing and implementing distributed systems a daunting task, partly because of the massive scale of typical deployments, the complexity and unpredictability of system executions, and also due to emerging security threats. System faults caused by design flaws, software bugs or vulnerabilities can result in catastrophic consequences. For example, a recent denial-of-service attack against DNS services paralyzed the Internet of the entire east coast. When a fault occurs, system operators often find themselves needing to *manually* perform a series of management tasks, such as root cause identification, system recovery and reconfiguration. Almost always, these tasks are time-consuming and tedious, even for experts. To address this challenge, my research adopts a multi-disciplinary approach by combining techniques from databases, security, networking and formal methods. **The unifying thrust of my research is the development of systematic approaches that empower system administrators to enhance the reliability and security of distributed systems in an automated manner.**

Towards this goal, my research on *provenance* introduces an approach that provides the fundamental functionality required for performing fault diagnosis and debugging—the capability to "explain" the existence (or change) of system state. Provenance is a form of metadata that tracks direct (and transitive) dependencies among system states and their changes. In distributed systems, system states are continually being generated, changed and moved. Such information is of great value; it permits system operators to tie observed faults to their potential root causes, and to assess the damage that these faults may have caused to the rest of the system. My research targets three themes: 1) provenance maintenance and querying for efficient and reliable dependency tracing; 2) secure forensics and verification that defend against security threats in adversarial environments; and 3) provenance-enabled automated fault diagnosis and debugging.

After joining Georgetown, my students and I have published across a diverse set of top-tier venues, including conferences in databases (VLDB [2, 21, 18], SIGMOD [10], CIDR [8]), networking (SIGCOMM [16, 19, 7], INFOCOM [13]), systems (OSDI [4], NSDI [17], Eurosys [3]), security (USENIX Security [1], ACSAC [14]), and formal methods (FORTE [9], PPDP [11]). In total, I have published 24 conference papers, seven journal papers, two books, and 18 other peer-reviewed demonstration and workshop papers. My research has led to two patent applications. My work has been supported by more than 2.9MM in grants and contracts, including four grants from NSF and a contract from the Defense Advanced Research Projects Agency (DARPA). I am also a recipient of the NSF CAREER award, the NSF's most prestigious award for junior faculty.

In the remainder of this statement, I present an overview of my recent research accomplishments and ongoing and future work. The full details of my research—including electronic copies of my publications—are available on my webpage at http://www.cs.georgetown.edu/~wzhou/.

Theme 1: Practical Provenance Support in Distributed Systems

My early research on Distributed Time-aware Provenance (DTaP) [21] provides a sound and complete provenance model that correctly captures the system dependencies, and a framework for maintaining and querying DTaP with low communication and computation cost. As DTaP captures detailed information about the entire execution, it natually faces concerns about the storage cost (e.g., in systems that handle high-volume streaming data), and the potential leakage of sensitive information (e.g., in systems that are managed by multiple administrative authorities). Since starting at Georgetown, I have significantly extended the work to address these critical challenges for the adoption of provenance in practical scenarios.

Provenance compression. The potentially large storage cost has been a long-standing challenge in provenance research. Provenance has to be incrementally maintained as network events occur. This becomes particularly challenging for the systems that deal with frequent and high-volume data packets. When there are streams of incoming events, the provenance information can become prohibitively large. In the "Distributed

Provenance Compression" paper (presented at SIGMOD 2017) [10], I explored techniques to dynamically compress distributed provenance stored at scale. The key insight is to apply static analysis on the specifications/protocols of a distributed system, and identify *equivalence classes* that group together input events sharing similar event triggering sequences. For example, in a routing system, two incoming packets with the same source and destination IP addresses will likely take the exact same route when traversing the network, as long as the routing tables do not change.

Based this observation, I defined an equivalence relation between provenance trees, and introduced a compile time static analysis phase for determining equivalence keys that can be used for grouping provenance trees together. At runtime, through inspection of the equivalence keys, the system maintains and stores only one concrete copy for provenance trees shared by all the members in the equivalence class. The experimental results demonstrated that this compression technique allows for significant—often orders of magnitude—storage reduction.

Privacy-preserving provenance. Another challenge in provenance deployment is the concern of data privacy. Provenance information collected on individual devices contains rich information that reveals details of system executions and decision making processes, some of which might be considered sensitive or confidential and should not be visible by unauthorized parties in the distributed system. When disseminating provenance information, we need to take into consideration privacy preferences of individual participants.

In the "Private Network Provenance" paper (to appear at VLDB 2017) [18], I introduced privacy-preserving provenance, a distributed provenance scheme that supports the richness of provenance while providing strong privacy guarantees over confidential data. I proposed a cryptographic approach to preserve the confidentiality of provenance (sub)graphs while allowing authorized nodes to query and access the parts that are visible to them. The proposed provenance framework leverages previous work on Searchable Symmetric Encryption (SSE) schemes. Informally, SSE builds a secure index (also called a dictionary) that maps each searchable keyword to the corresponding data item(s), but is only searchable by a keyword w if a user possesses a trapdoor for w. Even after the generated secure index is shipped to another entity, a node still controls the access of its data based on the roles of requesters. Privacy-preserving provenance guarantees that the only portions of the provenance graph that are revealed are those for which the querier is authorized. The empirical results demonstrated that it achieves this guarantee with negligible increases in latency and low bandwidth overhead.

Representative publications:

- Chen Chen, Harshal Tushar Lehri, Lay Kuan Loh, Anupam Alur, Limin Jia, Boon Thau Loo, and Wenchao Zhou. Distributed Provenance Compression. In *the 36th ACM SIGMOD International Conference on Management of Data (SIGMOD)*, May 2017. (included in the tenure package)
- Yuankai Zhang, Adam O'Neil, Micah Sherr, and Wenchao Zhou. Private Network Provenance. In *the 43rd International Conference on Very Large Databases (VLDB)*, Munich, Germany, August 2017. (included in the tenure package)

Theme 2: Forensics and Verification in Adversarial Environments

Provenance maintenance becomes significantly more difficult in adversarial environments. The adversary has the incentive and capability to cover his misbehavior and deny accusations against him; he can lie or even attempt to destroy evidence–such as provenance–against him. Additional assumptions and/or security treatment have to be introduced to ensure the integrity of the forensics collected during system execution, such that conclusions drawn from them are accurate and meaningful.

My early work on Secure Network Provenance (SNP) considered *completely untrusted* environments in which the adversary may have compromised an arbitrary subset of the nodes, and that he may have complete control over these nodes. I showed that, despite the conservative threat model, SNP still provides strong, provable guarantees for provenance maintenance and querying [20]: it ensures that an observable symptom of a fault or an attack can always be traced to a specific event—passive evasion or active misbehavior—on at least one faulty node, even when the adversary attempts to prevent this. Building on the SNP framework, I have further proposed primitives to support two important capabilities: to allow secure and efficient debugging of data-plane events, and to capture not only functional misbehaviors but also temporal misbehaviors.

Support for data-plane provenance. To achieve the strong guarantee, SNP introduces a suite of security protocols; as a result, it inevitably incurs high performance penalties when deployed on a distributed

system's data plane: keeping a comprehensive record of all packets that have ever traveled within the system would require a gigantic amount of storage. In the "Architectural Support for Internet Diagnostics" paper (presented at Eurosys 2017) [3], I showed that, by carefully applying optimizations such as periodic expiration and batching, and with the support of hardware optimized for parallel computation (such as NetFPGA), it is feasible to maintain secure provenance even at data plane. The proposed design provides a compact yet powerful primitive that can be used as the basic building block for supporting a wide range of Internet debugging tasks. Most importantly, it is practical: other than a small link-layer header, the design does not require any changes to the current data plane, and it uses only simple cryptographic primitives (mostly hashing) that can be implemented at line speed.

Time-deterministic replay. SNP constructs provenance graphs by replaying secure logs captured during runtime. However, deterministic replay reproduces the functional behavior of a program, but not its temporal behavior, which fundamentally limits the capability of identifying several classes of problems (e.g., race conditions or performance issues). The replayed execution can take substantially more (or less) time than the original execution, and the outputs can appear at very different points in both executions. In fact, this is a fundamental limitation of a majority of existing deterministic replay techniques.

To ensure that SNP captures not only a system's functional behavior, but also its temporal behavior (such that attacks related to timing, e.g., covert timing channels, can be detected), I introduced time-deterministic replay (TDR) [4], which reproduces the temporal behavior of events that have nondeterministic timing by carefully aligning its operations during play and replay so that they affect the program's timing in a similar way. In my work presented at OSDI 2014 [4], I demonstrated that TDR mitigated or eliminated many large sources of time noise; the timing of complex programs could be reliably reproduced on a commodity machine with an error of 1.85% or less.

Representative publications:

- Ang Chen, Andreas Haeberlen, Wenchao Zhou, and Boon Thau Loo. One Primitive to Diagnose Them All: Architectural Support for Internet Diagnostics. In *the European Conference on Computer Systems (EuroSys)*, April 2017.
- Ang Chen, W. Brad Moore, Hanjun Xiao, Andreas Haeberlen, Linh Thi Xuan Phan, Micah Sherr, and Wenchao Zhou. Detecting Covert Timing Channels with Time-Deterministic Replay. In *the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, October 2014. (included in the tenure package)

Theme 3: Automated System Repair and Synthesis

Provenance offers a causal chain of events that can link an effect to its root causes. But operators often wish not only to diagnose a problem (say, a software bug) but also to find a suitable fix. Together with my colleagues Andreas Haeberlen and Boon Thau Loo at the University of Pennsylvania, we pioneered the use of provenance for systematically generating potential repairs. For instance, in the network routing scenario, it is often difficult to determine the import/export policies and local preferences of potential routes. It would be helpful to have a systematic approach to guide administrators to answer questions like *"why is an expected system state not observed?"* and *"how can the system inputs or configurations be tuned to obtain a desired result?"*.

Negative provenance. Typical provenance models (such as the DTaP model) cannot (directly) handle these "why-not" and "how-to" questions: they require that the effect be observed in the execution and serve as a "lead" to start the reasoning; they cannot support reasoning of an effect (i.e., the expected system state) that never existed in the execution. Nevertheless, it is possible to construct a similar "backtrace" for negative events, using the concept of *negative provenance* [15, 16]. The key insight is to use counterfactual reasoning, i.e., to examine all possible causes that *could have* produced the missing effect. A top-down procedure is adopted for constructing the provenance of a given negative event "on demand", without materializing the entire negative provenance graph, which is typically infinite (e.g., containing an explanation for every tuple that could *potentially* exist). In the "Diagnosing Missing Events in Distributed Systems with Negative Provenance" paper (presented at SIGCOMM 2014) [16], I demonstrated that negative provenance is a useful tool in generating compact and readable results with the aid of post-processing heuristics.

Network diagnostics with differential provenance. As a further step towards automated network repair, the "Better network diagnostics with differential provenance" paper (presented at SIGCOMM 2016) [7] explores the potential of repairing misconfigurations through differential analysis across provenance trees.

The key observation is that network misconfigurations oftentimes only affect a subset of traffic/nodes, and they only manifest infrequently. Therefore, an operator typically has both working and non-working instances of similar traffic or service. In those cases, if we reason about the differences between the provenance trees for the working and non-working instances, we can likely identify the root cause of the problem. We call this approach differential provenance [6, 7].

One challenge in differential provenance is that a small, initial difference can lead to wildly different network executions afterwards, e.g., a different routing decision may send packets down an entirely different path. Therefore, the differences between working and non-working provenance trees can be much larger than one might expect. Differential provenance addresses this by rolling back the network execution to a point where the provenance trees start to diverge; then, it changes the mismatched tuple(s) on the non-working provenance tree to the correct version, and the rolls forward the execution until the two trees are aligned.

We have applied differential provenance in a number of case studies on repairing software-defined networks and Hadoop MapReduce jobs. Our results show that reasoning about the differences of provenance is quite effective: it can pinpoint one or two misconfigured entries to be the root cause of the problem

Automated program repair with meta provenance. When generating repairs, differential provenance only considers changes to configuration data, but not how computations are performed on the data. That is, it does not consider bugs in programs. In the "Automated Bug Removal for Software-Defined Networks" paper (presented at NSDI 2017) [17], I introduced Meta Provenance to reason about changes to both program and configuration data. Meta provenance has a set of meta tuples, which represent the syntactic elements of the program itself, and a set of meta rules, which describe the operational semantics of the programming language. Negative provenance is then applied to ask why some conditions did not hold. The result is a set of changes to the program and/or to the configuration data that makes the condition become true.

Once potential modifications are identified, we need to further examine each of them and evaluate whether it is actually a "correct" modification. That is, given the same system configurations and external inputs, the modified program should rectify faulty execution traces, and avoid altering the progress of execution traces that are previously correct. We leverage multi-query optimization from the database literature to speed up the backtesting, which can validate multiple repairs in a single run.

Representative publications:

- Yang Wu, Ang Chen, Andreas Haeberlen, Wenchao Zhou, and Boon Thau Loo. Automated Bug Removal for Software-Defined Networks. In *the 14th USENIX Symposium on Networked Systems (NSDI)*, March 2017
- Ang Chen, Yang Wu, Andreas Haeberlen, Wenchao Zhou, and Boon Thau Loo. The Good, the Bad, and the Differences: Better Network Diagnostics with Differential Provenance. In *the ACM SIGCOMM Conference on Data Communication (SIGCOMM)*, August 2016. (included in the tenure package)
- Yang Wu, Mingchen Zhao, Andreas Haeberlen, Wenchao Zhou, and Boon Thau Loo. Diagnosing Missing Events in Distributed Systems with Negative Provenance. In *the ACM SIGCOMM Conference on Data Communication (SIGCOMM)*, August 2014.

Ongoing and Future Work

Provenance, an artifact generated from system execution, often contains rich information that gives insights into the execution. Building upon the above research themes, I intend to further leverage the rich provenance data to enhance the development cycle of distributed systems.

Learning from provenance. Provenance can be viewed as an instantiation of the execution logic of a distributed system; the structural features mined from successful and failed executions may possess different features, which can be leveraged to discover system faults. This is analogous to the use of graph-based data mining in analyzing object usage in computer programs and debugging parallel program execution. In preliminary work [12], I have demonstrated that the structural features mined from provenance were useful to detect misbehavior such as prefix hijacking and black-holing in network routing. I plan to extend the work to broader application scenarios and improve its performance.

Provenance-assisted recovery. Another intriguing direction that I plan to explore is the use of provenance for *provenance-assisted* recovery. Provenance data retains sufficient information to reproduce the system execution trace individually for each node. With the support of time-deterministic replay, the replayed execution can even reproduce the exact timing. This brings opportunities to *undo* the damages caused by

an exposed system fault, by applying the inverse operations in the reverse order. For example, a mistakenly deleted system state can be restored by the corresponding insertion. In addition, provenance keeps the dependency information and thus allows *minimal recovery*, i.e., the recovery only impacts the nodes that are *actually* affected by the fault.

Provenance pertains useful information for diagnosing functional faults caused by design flaws, software bugs or misconfigurations. However, there is a large fraction of faults caused by other potential vulnerabilities, such as volumetric denial-of-service attacks and covert channel attacks. In the long term, my research goal is to understand and defend against these types of attacks, and facilitate the development of secure and reliable distributed systems. I briefly introduce two funded ongoing projects that pursue this goal:

Defense against distributed DoS attacks. One project that I am already exploring is to design a fundamentally different approach to defend against distributed denial-of-service (DDoS) attacks, which exhaust and paralyze network-based services by making repeated requests from multiple locations. DDoS attacks are notoriously difficult to defend against given the heterogeneity and distributed nature of the attack, rendering classic defenses such as filtering ineffective.

Services in today's clouds are deployed as part of monolithic applications and network stacks with resources provisioned for each layer in the stack. Attackers who can cause resource starvation at one layer can render the other provisioned resources unusable. For example, a TCP SYN flood attack consumes only memory at the TCP layer of the stack but renders unusable all resources dedicated to services that depend on formation of TCP connections. The problem with this structure is the interdependence and tight coupling of resources within the entire cloud stack. We resolve this by splitting the monolithic stack into many separable components called microservices. Each microservice may have its own decoupled set of resources, and the microservices themselves can be widely replicated on multiple containers. If part of the application stack is experiencing a DDoS attack, then we can then massively replicate just the affected components, potentially across many machines. This allows scaling of the impacted resource separately from the rest of the application stack, meaning that resources can be precisely added where needed to combat the attack. Preliminary work [5] has shown initial evidence that the microservice-based approach significantly outperforms naïve replication. Moving forward, I plan to further explore systematic and ideally automated procedure that identifies optimal separation of monolithic software into microservices.

Defense against covert channel attacks. Covert channel refers to an attack that creates a capability to transfer data through a medium that is not designed for data communication. With my colleagues at Georgetown University and University of California, Berkeley, I recently studied a special form of covert channel attacks called hidden voice commands [1]. The project demonstrated techniques for constructing audio (sounds) that are interpreted as voice commands by a computer speech recognition system, but are incomprehensible to human listeners. Well-constructed hidden voice commands could be embedded within videos on popular video sharing sites, or even broadcast over radio. While human listeners would not identify the commands, their nearby devices would.

The existing methods of creating hidden voice commands are manual processes that require non-automated parameter tuning and human testing to produce and then evaluate the human understandability of candidate commands. One direction is to redesign the engineering of hidden voice commands to be more automated and more effective. Specifically, I plan to introduce techniques for automating the production of hidden voice commands that correspond to arbitrary inputs.

In a parallel thread, I also plan to study the effectiveness of defense against hidden voice channels. For example, a potential and obvious defense against hidden voice commands is to perform speaker recognition, that is, a device only responds to queries and commands issued by an authorized voice. I plan to explore whether the techniques used to construct hidden voice commands necessarily hinder voice authentication.

References

[1] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. Hidden Voice Commands. In *USENIX Security Symposium (Security)*, 2016.

[2] Badrish Chandramouli, Suman Nath, and Wenchao Zhou. Supporting distributed feed-following apps over edge devices. In *the International Conference on Very Large Databases (VLDB)*, 2014.

[3] Ang Chen, Andreas Haeberlen, Wenchao Zhou, and Boon Thau Loo. One Primitive to Diagnose Them All: Architectural Support for Internet Diagnostics. In *European Conference on Computer Systems (EuroSys)*, 2017.

[4] Ang Chen, W. Brad Moore, Hanjun Xiao, Andreas Haeberlen, Linh Thi Xuan Phan, Micah Sherr, and Wenchao Zhou. Detecting covert timing channels with time-deterministic replay. In *the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2014.

[5] Ang Chen, Akshay Sriraman, Tavish Vaidya, Yuankai Zhang, Andreas Haeberlen, Boon Thau Loo, Linh Thi Xuan Phan, Micah Sherr, Clay Shields, and Wenchao Zhou. Dispersing Asymmetric DDoS Attacks with SplitStack. In *ACM SIGCOMM Hot Topics in Networks (HotNets)*, 2016.

[6] Ang Chen, Yang Wu, Andreas Haeberlen, Wenchao Zhou, and Boon Thau Loo. Differential Provenance: Better Network Diagnostics with Reference Events. In *ACM SIGCOMM Hot Topics in Networks (HotNets)*, 2015.

[7] Ang Chen, Yang Wu, Andreas Haeberlen, Wenchao Zhou, and Boon Thau Loo. The Good, the Bad, and the Dierences: Better Network Diagnostics with Dierential Provenance. In *Proceedings of ACM SIGCOMM Conference on Data Communication (SIGCOMM)*, 2016.

[8] Ang Chen, Yang Wu, Andreas Haeberlen, Wenchao Zhou, and Boon Thau Loo. Data Provenance at Internet Scale: Architecture, Experiences, and the Road Ahead. In *Proceedings of Conference on Innovative Data Systems Research (CIDR)*, 2017.

[9] Chen Chen, Limin Jia, Hao Xu, Cheng Luo, Wenchao Zhou, and Boon Thau Loo. A program logic for verifying secure routing protocols. In *the International Conference on Formal Techniques for Distributed Objects, Components and Systems (FORTE)*, 2014.

[10] Chen Chen, Harshal Tushar Lehri, Lay Kuan Loh, Anupam Alurand Limin Jia, Boon Thau Loo, and Wenchao Zhou. Distributed Provenance Compression. In *Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2017.

[11] Chen Chen, Lay Kuan Loh, Limin Jia, Wenchao Zhou, and Boon Thau Loo. Automated Verication of Safety Properties of Declarative Networking Programs. In *Proc. Int. ACM SIGPLAN Symp. on Principles and Practice of Declarative Programming*, 2015.

[12] David DeBoer, Wenchao Zhou, and Lisa Singh. Using substructure mining to identify misbehavior in network provenance graphs. In *the First International Workshop on Graph Data Management Experience and Systems (GRADES)*, 2013.

[13] Calvin Newport and Wenchao Zhou. The (surprising) computational power of the sdn data plane. In *the IEEE Conference on Computer Communications (INFOCOM)*, 2015.

[14] Jordan Wilberding, Andrew Yates, Micah Sherr, and Wenchao Zhou. Validating web content with senser. In *the Annual Computer Security Applications Conference (ACSAC)*, 2013.

[15] Yang Wu, Andreas Haeberlen, Wenchao Zhou, and Boon Thau Loo. Answering why-not queries in software-defined networks with negative provenance. In *the 12th ACM Workshop on Hot Topics in Networks (HotNets-XII)*, 2013.

[16] Yang Wu, Mingchen Zhao, Andreas Haeberlen, Wenchao Zhou, and Boon Thau Loo. Diagnosing missing events in distributed systems with negative provenance. In *the ACM SIGCOMM Conference on Data Communication (SIGCOMM)*, 2014.

[17] Ang Chen Yang Wu, Andreas Haeberlen, Wenchao Zhou, and Boon Thau Loo. Automated Bug Removal for Software-Dened Networks. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2017.

[18] Yuankai Zhang, Adam ONeil, Micah Sherr, and Wenchao Zhou. Private network provenance. In *the International Conference on Very Large Databases (VLDB)*, 2017.

[19] Mingchen Zhao, Wenchao Zhou, Alexander J.T. Gurney, Andreas Haeberlen, Micah Sherr, and Boon Thau Loo. Private and verifiable interdomain routing decisions. In *the ACM SIGCOMM Conference on Data Communication (SIGCOMM)*, 2012.

[20] Wenchao Zhou, Qiong Fei, Arjun Narayan, Andreas Haeberlen, Boon Thau Loo, and Micah Sherr. Secure Network Provenance. In *ACM Symposium on Operating Systems Principles (SOSP)*, 2011.

[21] Wenchao Zhou, Suyog Mapara, Yang Li, Andreas Haeberlen, Zachary Ives, Boon Thau Loo, and Micah Sherr. Distributed time-aware provenance for distributed systems. In *the International Conference on Very Large Databases (VLDB)*, 2013.