My research focuses on building resilient systems across a range of architectures, including enterprise systems and cloud computing. More specifically, I consider the question of proving the integrity of systems and the data being processed. Confidentiality is often the first security guarantee considered when building secure systems. Integrity is another key building block for secure systems, and is challenging to ensure, as it requires deep understanding of how a system operates, and how that integrity is impacted by inputs from the outside world. My work has identified new research thrusts that address national cyber security issues. These experiences are the driving force of a research agenda that is grounded in real world threats to national security.

Secure Data Provenance

After joining MIT Lincoln Laboratory (MITLL), I have focused on problems faced by decision makers within the government. I learned that they rely heavily on data to make critical decisions, and the systems that process this data must be high-integrity. Today, they place trust in the systems that process data with no basis for that trust. With that knowledge, I began exploring ways to provide decision makers with timely and accurate data to ensure mission success. Achieving a high-integrity system is challenging, as it requires knowing what constitutes "good" system operation.

While at MITLL, I have had the fortune of remaining active in academic research, publishing in top security venues, including the USENIX Security Symposium. This has been made possible in part by mentoring graduate and undergraduate students as part of MITLL's summer research program. I have worked with students from a number of universities, often continuing collaborations beyond the summer. These collaborations have resulted in several conference and workshop publications. One student has recently graduated and received an appointment at the University of Illinois Urbana-Champaign as an Assistant Professor.

My research focuses on building systems that can collect and report on the integrity of the system and the data being processed, providing strong security guarantees for the integrity "proofs" generated by these systems, broadly divided into three major thrusts. The first area examines how to secure data as it is processed using *data provenance*, or the history of ownership and processing of data. Next, I look at how to secure systems using *integrity measurement* and *trusted computing* when the system in question is physically controlled by the user, for example a traditional enterprise server or desktop system. Finally, *secure cloud computing* requires a change in thinking, especially in cases where access to trusted hardware is limited.

Data provenance, also known as lineage or pedigree, tracks the evolution of data as it is processed. Provenance is often compared to taint tracking, but differs in that provenance describes not only *w*hat inputs contributed to the current state but *how* the data was processed and *who* contributed to the computation. Provenance is not a new concept, having roots in the fine art community, where provenance records for a piece of artwork are used to determine the authenticity of the work.

Using data provenance to secure a system requires a number of steps, including collection, encoding, storage, analysis, and adaptation. Collection can be done in a number of different locations within the system, including within the operating system itself, but must be done securely. We developed Linux Provenance Modules (LPM), a framework for building provenance modules into the operating system with strong security guarantees about the collected data [3]. This work defines the security requirements for any secure whole-system provenance collection mechanism. This is the first fully-realized *provenance monitor*, a system that provided guarantees as outlined in the 1972 Anderson report that identifies the requirements for a reference monitor. This system demonstrates the mechanisms required to support distributed provenance through a cryptographic commitment protocol for exchanging data between provenance-aware nodes. The LPM work also explores a mechanism to create a data-loss prevention system that prevents the spread of sensitive data. Extensive benchmarking revealed overheads as low as 2.7%, and an ability to determine the ancestry of a system object in milliseconds, fast enough to support on-line queries for provenance of system objects.

During the benchmarking of LPM, we realized that whole-system provenance generates large volumes of data, a problem that can be addressed in different ways depending on the target use case for provenance. We have explored different mechanisms to deal with this volume of provenance data, including filtering the provenance based on what *can* impact critical data. In order to filter the provenance, we first identified a reliable mechanism to label applications and data, and determine if a particular application, data object, or information flow can influence the operation of the identified applications. One mechanism to do this is to analyze the information flow policy of the application. ProvWalls is a system that analyzes the MAC policy of an application to determine allowed data flows [1]. Flows that are allowed to flow either directly or indirectly into the critical application are added to a policy that defines what provenance records to keep. This policy is loaded into the running system and checked before provenance is recorded. Initial experiments with this system show significant reduction in storage overheads for some workflows, up to 82% in some workloads [1].

The downside to filtering arises when provenance queries are not known *a priori*. Provenance deemed irrelevant today may in fact be critical tomorrow. Additionally, identifying what system activity can impact data is a non-trivial task. I explored mechanisms to efficiently store and analyze the provenance data for cases where filtering is not possible. Using Apache Accumulo[1] and D4M,[2] we demonstrated an approach to efficiently ingest system provenance data and build the graphs necessary to analyze the provenance captured by the system [8]. Using Accumulo and D4M, we demonstrated a system that supports ingest rates of over 4,000 provenance events per second, supporting peak workloads for system provenance. The system is able to store and analyze all of the system provenance, supporting the widest possible range of queries.

While tracking provenance at the system layer provides strong security guarantees about the collected data, and has the desirable property that applications do not require modifications to run, a *semantic gap* exists between the execution of an application, i.e. the business logic of the application, and how that interaction manifests at the OS layer. For example, a relational database like MySQL has notions of tables, rows, columns, cells, etc. From the system perspective, MySQL is interacting with files on disk using traditional read and write system calls for every database operation. The semantics of the database are lost, with no easy way to reconstruct them for later analysis. To address this problem we explored Database-Aware Provenance (DAP), a system that bridges the semantic gap without modifying each application to capture provenance [2]. Instead, DAP parses the protocols used to interface with these applications to capture workflow provenance that accurately describes the high-level operations of an application. The prototype targets an SQL database, capturing the tables and columns read and written by applications interfacing with the database. We then demonstrate a guard mechanism that queries the provenance store to determine the data being sent to the client by the web server. If the provenance indicates sensitive data is included in the response, but should not be, the request is blocked, preventing data exfiltration attacks. This additional query and analysis introduces only six milliseconds of latency for each client request.

After exploring several mechanisms to integrate provenance into systems, it became evident that data provenance for system security required an ecosystem of tools and libraries for software developers and system engineers. In [7], we provide an overview of the tools, libraries, and systems for capturing and leveraging provenance to build resilient systems[3]. The survey describes tools available to developers to integrate data provenance into their system. We then identify next steps that are required for integrating data provenance into systems and applications. Finally, we explore the integration of secure end-to-end provenance into a prototype mission system for logistics planning, highlighting the remaining research challenges in secure data provenance, outlining a roadmap for future provenance research.

Integrity Measurement and Trusted Computing

In the same way that a building is only as solid as its foundation, data processed on a low-integrity system should be considered low-integrity. Determining the integrity of a system requires *integrity measurement*, which answers the question "What is the current integrity state of the system?" This question is challenging to answer, but provides a solid foundation for secure data processing. This is a question that has plagued me since I began graduate school, and continues to this day.

During system operation, measurements are taken that reflect the current integrity state. These measurements must be protected against malicious modification. This protection is a *root of trust* that forms the basis for assessment of the systems trustworthiness. One such commonly used root of trust is the *Trusted Platform Module*, or TPM. The

[1]See: `https://accumulo.apache.org`

[2]See: `http://www.mit.edu/~kepner/D4M/`

[3]A resilient system is one that can "fight through" an attack and still achieve its stated mission, maybe in a degraded manner.

TPM is a low-cost alternative to more expensive secure co-processors (e.g. the IBM 4758 and successors) and is nearly ubiquitous in commodity systems today. However, the TPM is not without downsides.

One downside to the TPM is the speed of some critical operations, such as digital signatures. Signing is a key step in generating an *attestation* that reports the current integrity state of the system. Generating this signature takes anywhere from 500 milliseconds to 1 second, and is done serially if more than one attestation is requested at the same time. Clients connecting to a remote system use trusted computing protocols to request an attestation from the remote machine to validate the current state of the system before entrusting that system with critical data. With many clients requesting attestations, the latency induced per client quickly reaches unreasonable levels due to the single-threaded nature of the signing process. I created the *Spork* system that amortizes the cost of the attestation across multiple client requests while still providing strong guarantees about the attestation [6]. Additionally, Spork binds the content being served to the integrity proof, ensuring that the content and the proof cannot be served by two separate systems to hide the fact that a system has been compromised.

While the Spork system works well for "Web 1.0"-style systems where content is static, it does not account for dynamic content that is typical in modern "Web 2.0"-style applications. Applying Spork to asynchronous requests for data introduces an unacceptable latency for each request. To address the latency introduced by Spork, I developed the *Sporf* system that explores novel cryptographic constructions that allow the system to provide similar security guarantees with lower latencies required by modern web applications [9]. Pre-computation of one-time keys that are bound to the attestation are used to sign the data being sent to the client. This further amortizes the cost of the signing operations. With offline-online signatures, Sporf is able to sign dynamic content in a little as 200 milliseconds, retaining the responsive nature of modern web applications. With additional hardware support, I show how to reduce that latency even further to 81 milliseconds.

At the same time that I was exploring ways to secure web applications, I also realized that the systems hosting these applications required fine-grained integrity protection at the lowest levels. A trend in storage devices enhances the capability of the storage controllers to include additional functionality, a feature that I explored to build secure systems "from the ground up". We proposed policy enforcement mechanisms that make access decisions as close to the data as possible, namely within the disk itself. In [4], we demonstrated the *SwitchBlade* architecture that provides strong isolation primitives using physical tokens to enable and disable access to segments of the disk.

Continuing the trend of "security from the bottom up", I considered ways to provide integrity from the first bytes written to a new system, i.e. the operating system itself. A *root of trust for installation*, or ROTI, binds the installation media to the installed system, providing a mechanism to validate the origin of critical boot files for the system, such that a verifier can validate the source of the installation and ensure that the system has not been modified. In [11], we construct a network-based ROTI, netROTI, building on prior work that constructed a ROTI from physical installation media. With the advent of cloud computing, installations are done via a network, and not via physical media, necessitating the migration from physical media-based ROTIs to a network-based ROTI. The netROTI provides strong guarantees about the physical host system, and the origin of critical boot files on the installed system for network-based installations.

Secure Cloud Computing

The DoD is currently considering the best way to leverage cloud computing services, either frompopular cloud providers like Amazon, Google, and Microsoft, or by consolidating resources and hosting a private cloud. However, this transition from "in-house" computing to cloud computing is fraught with security issues. My research has focused on ways to support this transition while providing a secure foundation for cloud computing. There are many questions that remain unanswered when considering how to securely migrate to a cloud environment. Users now rely on pools of shared resources, where they don't always know who the other users are. This change to shared resources leads to problems like reduced visibility into the state of the system. What is needed is a way to leverage integrity measurement and trusted computing building blocks to build high-integrity cloud instances, taking into account the lack of access to the physical hardware that form the roots of trust for high-integrity systems as described above.

One question I began exploring when considering the move to the cloud is "What is the integrity state of my cloud instances, and how can I be sure the host is not lying about it?" With a traditional server, TPMs and integrity measurement allow a user to validate the current state of their system, with roots in secure hardware. With a virtual machine, physical hardware access is challenging, especially for a special-purpose device that is not designed to be multiplexed between virtual machines, i.e. the TPM. Instead, we leveraged policy enforcement and verification that the *physical host* is enforcing the desired integrity policies. The host level verification is done using the methods described in the previous section, and cloud instances are validated through the Virtual Machine Verifier (VMV). The VMV builds a system that enforces a specific integrity policy for a virtual machine, and generates proofs that the client's policy is being enforced [12].

As we experimented with the VMV, it became clear that scalability of the VMV approach would be limited by the number of clients connecting to the system. Instead of clients validating the integrity of each node, a centralized monitor validates the cloud instances, and takes appropriate corrective action when integrity violations are detected. The client validates the integrity of the monitor node by requesting an attestation, and uses this attestation as the root of a chain of trust for validating the instances running in the cloud [13].

It isn't enough to validate the code running on the system. Another challenge that arises in cloud computing is the bootstrapping of the necessary secrets, such as the public key used to identify a particular instance. Today, this is done by handing the private key to the cloud provider and letting the provider infrastructure pre-position the key during the provisioning process[4]. Using this approach, the user is trusting the provider not to steal that key and use it to masquerade as the node. In the *Keylime* system, we leverage the TPM and Virtual TPMs (software-based TPMs that are backed by a physical TPM and isolated to prevent tampering) to securely bootstrap identity keys in cloud nodes while minimizing the trust placed in the provider to securely handle secrets destined for the client nodes [10]. Another aspect of this approach that I explored was the exact method to integrate the trusted computing identity of the system with the the higher level applications that require cryptographic keys. Instead of patching each application to understand the Trusted Computing Group protocols, I showed a method for binding traditional cryptographic keys (i.e. keys not protected by the TPM) to the TPM-based identity key that is necessary to uniquely identify the node, while hiding the complexity of the TPM from the users.

While the majority of computation is done on the cloud nodes, some modest amount of computation may be done on the local system. As such, policy enforcement becomes more challenging in cloud environments, where services distributed across instances in the cloud and at the client end-points. *End-to-end policy enforcement* for a distributed system includes enforcement on each of these nodes. *FlowwolF* is a distributed system that enforces an end-to-end access control policy, using a web server and browser as the canonical example, but is extensible to other systems [5]. This work provides a foundation for secure cloud computing that process high-integrity data. However, more work is needed to build secure and trustworthy applications.

Future Work

My future work will expand on each of the themes above, broadly focusing on systems security. I plan to look for cross-cutting challenges that enable collaboration with the broad goal of building high-integrity systems that ensure data is processed in a secure manner in order to provide decision makers with timely and accurate data. My main thrust will be to continue to research secure data provenance, looking for new ways to securely collect and process provenance data. As we develop new mechanisms to secure collect provenance, I will explore ways to exploit the provenance to build a wide-range of high-integrity systems, including secure cloud computing, autonomous systems, and the Internet of Things, all challenges identified by the White House, the DoD, and industry as priorities in research roadmaps.

Provenance is a rich-data source for securing systems and determining how to integrate provenance into a range of systems is a continuing challenge. Integrating secure data provenance into systems to automate defenses requires new analysis techniques to process and respond to provenance as it is reported by the system. These response capabilities will allow a system to "fight through" an attack, not just block the attack at the expense of the desired system functionality. Additionally, as cloud computing continues to be used for more and more services, the need for new security architectures and enforcement mechanisms continues to increase. Systems like *Keylime* only scratch the surface of building secure applications in the cloud. I plan to continue to explore mechanisms to securely provision, monitor, and enforce policy in the cloud in ways that are as transparent to the end users as possible, ensuring a path towards wide-spread adoption. Measuring the integrity of a system as it operates will remain a challenge, with

[4]This is done using tools like `cloud-init`, a widely used provisioning tool for cloud nodes.

new techniques needed to ensure that an application remains high-integrity as it runs. Secure cloud computing is necessary to support other computing architectures that rely on the ability to outsource computation.

A more recent trend is the "Internet of Things" (IoT) that is rapidly connecting more and more devices to the internet. These resources limited edge devices collect data and do some limited processing, relying on large centralized infrastructures to aggregate and analyze data. As more and more IoT devices are connected, cloud computing will connect IoT devices to aggregation and analysis services. The security of these devices is a major concern, both within in industry and within the DOD. Traditional approaches to securing systems may be too heavyweight for resource limited devices, requiring new lightweight security primitives to provide the secure foundation for a secure Internet of Things. One area that I plan to explore in the near-term is the application of secure data provenance to this now highly distributed environment to protect the integrity of the data from end-to-end. Another is monitoring and enforcing system integrity for these devices.

Finally, as systems become more and more complex, it becomes increasingly difficult for operators to adequately secure the systems against *all* attacks using traditional defenses. Instead, security functions must have a degree of autonomy to ensure that systems can continue to operate through attacks, even in the face of new and ever-changing adversary tactics. The idea of autonomous security is one that requires careful design and consideration, and will rely on recent advances across a range of areas. This is a growing area of interest to many within the DoD and across other communities as well.

Secure systems require strong foundations to establish and maintain trust in the system and the data. Integrity plays a vital role in that foundation, and my work examines numerous ways to establish a high-integrity foundation both in traditional enterprise environments (i.e. client-server systems) and in cloud computing environments using integrity measurement and trusted computing. Data provenance extends integrity guarantees to the data being processed by these high-integrity systems. In the future, I plan to expand my focus to other systems and new techniques that integrate with my existing work to provide strong foundations for high-integrity systems.

Funding Opportunities

To support the above research agenda, I plan to pursue funding from a number of sources. The NSF Secure and Trustworthy Cyberspace (SaTC) program is one funding opportunity that is directly aligned with the research interests above. I also plan to watch for upcoming DARPA I2O programs that align with my interests. Example of current and past programs within I2O include the recently started Transparent Computing and Mission-oriented Resilient Cloud programs. Given my experience at MITLL, I feel I am also well-positioned to pursue funding opportunities within the Department of Defense, e.g. the Army Research Office and the Office of Naval Research (ONR). These funding opportunities will support the research agenda outlined above, and expand our understanding of systems and how they can be made resilient to attack.

References

[1] Adam Bates, Kevin R.B. Butler, and Thomas Moyer. "Take Only What You Need: Leveraging Mandatory Access Control Policy to Reduce Provenance Storage Costs". In: *7th USENIX Workshop on the Theory and Practice of Provenance (TaPP 15)*. Edinburgh, Scotland: USENIX Association, July 2015. URL: `https://www.usenix.org/conference/tapp15/workshop-program/presentation/bates`.

[2] Adam Bates, Kevin Butler, Alin Dobra, Brad Reaves, Patrick Cable, Thomas Moyer, and Nabil Schear. "Retrofitting Applications with Provenance-Based Security Monitoring". `https://arxiv.org/abs/1609.00266`. Sept. 2016.

[3] Adam Bates, Dave Tian, Kevin R.B. Butler, and Thomas Moyer. "Trustworthy Whole-System Provenance for the Linux Kernel". In: *24th USENIX Security Symposium (USENIX Security 15)*. Washington, D.C.: USENIX Association, Aug. 2015. URL: `https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/bates`.

[4] K. Butler, S. McLaughlin, T. Moyer, and P. McDaniel. "New Security Architectures Based on Emerging Disk Functionality". In: *IEEE Security Privacy* 8.5 (Sept. 2010), pp. 34–41. ISSN: 1540-7993. DOI: `10.1109/MSP.2010.90`.

[5] Boniface Hicks, Sandra Rueda, Dave King, Thomas Moyer, Joshua Schiffman, Yogesh Sreenivasan, Patrick McDaniel, and Trent Jaeger. "An Architecture for Enforcing End-to-End Access Control Over Web Applications". In: *Proceedings of the 2010 Symposium on Access Control Models and Technologies, SACMAT '10*. 2010.

[6] T. Moyer, K. Butler, J. Schiffman, P. McDaniel, and T. Jaeger. "Scalable Web Content Attestation". In: *IEEE Transactions on Computers* 61.5 (May 2012), pp. 686–699. DOI: `10.1109/tc.2011.60`. URL: `http://dx.doi.org/10.1109/TC.2011.60`.

[7] Thomas Moyer, Patric T. Cable, Karishma Chadha, Robert Cunningham, Nabil Schear, Warren Smith, Adam Bates, Kevin Butler, Frank Capobianco, and Trent Jaeger. "Leveraging Data Provenance to Enhance Cyber Resilience". In: *1st IEEE Cybersecurity Development (SecDev)*. 2016.

[8] Thomas Moyer and Vijay Gadepally. “High-throughput Ingest of Data Provenance Records into Accumulo”. In: *2016 IEEE High Performance Extreme Computing Conference, HPEC*. 2016.

[9] Thomas Moyer, Trent Jaeger, and Patrick McDaniel. “Scalable Integrity-Guaranteed AJAX”. In: *Proceedings of the 14th Asia-Pacific Web Conference (APWeb)*. Kunming, China, Apr. 2012.

[10] Nabil Schear, Patric T. Cable II, Thomas Moyer, Bryan Richard, and Robert Rudd. “Bootstrapping and Maintaining Trust in the Cloud”. In: *Proceedings of the 32nd Annual Computer Security Applications Conference*. ACSAC 2015. Los Angeles, CA, USA: ACM, 2016, pp. 1–10. ISBN: 978-1-4503-3682-6. DOI: `10.1145/2818000.2818003`. URL: `http://doi.acm.org/10.1145/2818000.2818003`.

[11] Joshua Schiffman, Thomas Moyer, Trent Jaeger, and Patrick McDaniel. “Network-based Root of Trust for Installation”. In: *IEEE Security & Privacy Magazine* (Jan. 2011).

[12] Joshua Schiffman, Thomas Moyer, Christopher Shal, Trent Jaeger, and Patrick McDaniel. “Justifying Integrity Using a Virtual Machine Verifier”. In: *Proceedings of the 2009 Annual Computer Security Applications Conference, ACSAC ’09*. Dec. 2009.

[13] Joshua Schiffman, Thomas Moyer, Hayawardh Vijayakumar, Trent Jaeger, and Patrick McDaniel. “Seeding Clouds with Trust Anchors”. In: *CCSW ’10: Proceedings of the 2010 ACM workshop on Cloud computing security*. ACM, 2010.