RESEARCH STATEMENT

Kangkook Jee (kjee@nec-labs.com)

We are in an era when data represents values, and information security plays a key role in guarding the flow of data. Thus, high security of the information infrastructure deeply relates to our everyday lives with a growing need to monitor quality and accuracy. A primitive system that accurately tracks the flow of a given piece of information is an inevitable requirement in building an accountable and traceable information infrastructure. Throughout my graduate study and years at NEC as a security researcher, my primary research has centered on the system-level support for information flow tracking (IFT). I also expanded this research to general system security to address the diverse security problems related to accuracy, efficiency, and scalability as core design principles. I define myself as a security system builder with a well-rounded understanding of attackers mindset and their attack vectors. I hope to continue to extend my experience as a system security researcher to be better prepared for the coming years of information technology conversions. As for my research agenda, I want to enhance my research on end-host oriented security by seeking higher-quality signals to help administrators make better security decisions. I also want to advance the automation of security analysis through machine learning (ML) algorithms. Finally, I want to explore various security issues in the emerging domain of IoT (*e.g.,* automotive, home automation).

System-level Support for IFT

My graduate thesis mainly focuses on implementing process-level support for IFT, with the primary design goals of accuracy, reusability, and efficiency. For my initial research prototype Libdft [7], my team chose *dynamic* hypervisor-based instrumentation. The design choice allowed the framework to run and analyze any arbitrary binary file along with its dependent libraries. Libdft explored different approaches for implementing efficient instruction-level data tracking, introduced a performant and 64-bit capable shadow memory, and identified the common pitfalls responsible for the excessive run-time overhead of similar tools. However, the cost of using runtime instrumentation to interleave information tracking logic along with the original execution was high; thus, it made Libdfts approach impractical for everyday operations. To address the runtime overhead issue, I proposed new optimizations that work against information tracking logic to reduce the number of operations. This approach first extracts tag propagation logic from all of the machine instructions and expresses those into newly proposed intermediate representation [6] so we can apply compiler optimization techniques to eliminate redundant operations.

To further expand on this work, I implemented a parallelization framework for IFT, ShadowReplica [5]. ShadowReplica extracted the information tracking logic from the original execution to run it in parallel with different CPU cores. While the research needed to use a variety of advanced program analysis techniques, the major challenge was to establish an efficient communication channel between cores considering hardware-specific aspects such as CPU cache hierarchy, and cache line sharing latency. By combining both optimization techniques, the approach my team used could achieve 4× more speedup over our base-line IFT framework.

IntFlow [9] took the *static* instrumentation approach by interleaving the tracking logic along with the source code at the compile time. Even with its limited coverage, IntFlow could achieve a similar level of accuracy with lower runtime overhead. Thus, I designed and implemented IntFlow [9], which addressed the issue of too many false alerts of the integer error checkers. IntFlow tracked the source of the integer operations only to focus on the integer errors affected by the external sources.

Enterprise Security with End-host Monitoring

Our computer security team at NECLA dedicated years to prototyping a next-generation security solution to address the known limitations of the conventional security solutions. The research prototype, automated security intelligence (ASI) [1], deploying the data collection agent program to each end-host to the collection-system call-level system activities. The agent streams collected information to the backend host for a number of advanced security analyses. We successfully productized and deployed the solution [1], and the product is currently deployed to multiple sites with hundreds of machines reporting their activities to the backend.

With the aim of becoming the main solution for enterprises, the size of deployment often comes in orders of thousands or tens of thousands; therefore, the foremost concern was to address the scalability of the product. ASI takes system call-level activities as its primary input, which are coarser than instruction-level system operations. Yet, each end-host from our deployment reported, on average, 1GB of system activities per day, and this posed a typical big data problem whose demand for backend storage exceeds common system design conventions. Since support for the forensic analysis is one of the products main features, it is required to store collected data for a long duration. Based on the data access pattern that we could observe from our deployment, I proposed a number of data reduction schemes [12, 11]. These schemes implemented the compression algorithms with the aim of saving a significant amount of storage at the cost of data loss without affecting the accuracy of security analyses. The reduction could save, on average, 30× data storage and up to 70× for specific applications. Another interesting research topic has been finding cost-efficient data sources, i.e., information that would maximize the benefit of security analysis while keeping storage and processing requirements quite small. Thus, I explored per-process DNS. By associating each process with its DNS queries and related meta-information (e.g., domain WHOIS, IP WHOIS, geo-locational information), we could significantly improve the monitoring visibility and the detection accuracy [10].

One security application that my team built on top of ASI streaming is graph-based machine learning (ML) detection which labels any previously unseen system activities as an anomaly. The security community considers this a promising research direction that would accelerate the system event investigation process by indicating previously unseen anomalies. However, ML-based detection also introduces another problem of "alert explosion" since it would label all of the previously unseen events as anomalies. It is still a difficult task to differentiate real malicious attacks from benign statistical outliners.

To tackle this issue, my team proposed an information flow-based approach that restores the context around reported alert [4] to make graph-level comparisons to previously confirmed true alerts. Upon detection of security of an anomaly, ASI prepared APPs for forensic analysis. To facilitate data lookup and navigation, we designed query languages to support the attack investigation. The AIQL [3] query language works for event data stored in the database, and SAQL [2] is a stream query language that works for the event stream, both of which allow users to write their custom detection rules. Casualty analysis [8] tracks multi-hop causal relationships among system resources (e.g., file, process, network socket) to diagnose attack provenances and consequences. This is another way to implement IFT. The analysis reconstructs dependency among coarsely grained system resources. ASI also restores information flow across the system boundary connecting network end-points among hosts inside the enterprise. Since the size of the dependency graph often grows exponentially, to facilitate timely analysis, I proposed a number of additional reduction schemes [8].

Research Agenda

In the coming years, I plan to work on the following research topics.

More end-host security. Security solutions are continually needed for a robust feature that captures detailed system evidence and is difficult for attackers to manipulate. While an end-host solution can query underlying systems for such detailed information, accessing the feature in an efficient and reliable manner is another system challenge. One of the system events on my wish list is the URI request and its responsible process (URI-to-Process). Up-to-date security solutions monitor the DNS name or IP address at host granularity, and the ASI-associated DNS name and its responsible processes. This URI-to-Process information reveals query strings that are essential for understanding the program behavior; thus, they provide the proper basis for accurate detection. URI information is typically difficult to extract when the connection is encrypted, and we see more traffic being encrypted (e.g., DNS-over-HTTPS) for privacy reasons. By having direct access to the system, I would like to explore how to extend the end-host security sensor to extract such information in plaintext.

ML-assisted security automation. I would also like to explore ways to scale security analysis with ML-assisted automation. End-host security collects detailed system activities from each endpoint. The amount of information as well as the number of investigation cases will only increase. Security is one area that has few experts and is in high need of automation to help security investigators solely focus their time on critical new problems. Although it is a long research trail, the attempts to implement ML-based security detection for low-level system information have been mostly unsuccessful. Individual system events have tended to be too isolated, which makes it difficult to restore meaningful semantics when a few events are co-related. To improve this issue, I would like to group a sequence of events to represent a meaningful unit. An event sequence can represent behaviors such as an external IP connection without DNS resolution, and deletion of the binary after its execution. Such pre-processing of a system event can improve the ML algorithm for both training and prediction. Since perfecting detection techniques may well be a far-fetched goal, I would prefer to focus on maximizing automation with a minimal requirement for human guidance.

End-host Security for IoT environment. I want to explore end-host security for IoT where each edge devices computation resources are severely constrained. We have seen increasing attacks from the IoT environment, and conventional security practices no longer hold in this domain. In addition, the IoT devices resource allowance for security is quite limited, so we also cannot assume a stable network for mobile IoT devices (*e.g.,* automobiles). On the other hand, IoT devices are often dedicated to specific tasks with more static and predictable usage patterns. Thus, it is easier for an ML algorithm to build a more stable model for security analysis.

The efficient summarization and coordination of the network connection between the edge device and the cloud is another interesting research problem. One of my design proposals separates the expensive task of an ML model update from the edge device and delegates it to the cloud. This approach would minimize the computation requirement imposed on the edge, so expensive operations such as model updates and maintenance will occur only from the cloud. So far, I led a research project with a prototype of advanced security solutions for an IoT and automobile environment. I hope to continue this work to deliver an impactful outcome that would advance IoT security.

References

[1] Automated Security Intelligence (ASI). `https://www.nec.com/en/global/techrep/journal/g16/n01/160110.html`, 2018.

[2] Gao, P., Xiao, X., Li, D., Li, Z., Jee, K., Wu, Z., Kim, C. H., Kulkarni, S. R., and Mittal, P. SAQL: A stream-based query system for real-time abnormal system behavior detection. In *USENIX Security Symposium* (2018), USENIX Association, pp. 639–656.

[3] Gao, P., Xiao, X., Li, Z., Xu, F., Kulkarni, S. R., and Mittal, P. AIQL: enabling efficient attack investigation from system monitoring data. In *USENIX Annual Technical Conference* (2018), USENIX Association, pp. 113–126.

[4] Hassan, W. U., Guo, S., Li, D., Chen, Z., Jee, K., Li, Z., and Bates, A. Nodoze: Combatting threat alert fatigue with automated provenance triage. In *NDSS* (2019), The Internet Society.

[5] Jee, K., Kemerlis, V. P., Keromytis, A. D., and Portokalidis, G. Shadowreplica: efficient parallelization of dynamic data flow tracking. In *ACM Conference on Computer and Communications Security* (2013), ACM, pp. 235–246.

[6] Jee, K., Portokalidis, G., Kemerlis, V. P., Ghosh, S., August, D. I., and Keromytis, A. D. A general approach for efficiently accelerating software-based dynamic data flow tracking on commodity hardware. In *NDSS* (2012), The Internet Society.

[7] Kemerlis, V. P., Portokalidis, G., Jee, K., and Keromytis, A. D. libdft: practical dynamic data flow tracking for commodity systems. In *VEE* (2012), ACM, pp. 121–132.

[8] Liu, Y., Zhang, M., Li, D., Jee, K., Li, Z., Wu, Z., Rhee, J., and Mittal, P. Towards a timely causality analysis for enterprise security. In *NDSS* (2018), The Internet Society.

[9] Pomonis, M., Petsios, T., Jee, K., Polychronakis, M., and Keromytis, A. D. Intflow: improving the accuracy of arithmetic error detection using information flow tracking. In *ACSAC* (2014), ACM, pp. 416–425.

[10] Sivakorn, S., Jee, K., Sun, Y., Kort-Parn, L., Li, Z., Lumezanu, C., Tang, L.-A., and Li, D. Countering malicious processes with process-dns association. In *NDSS* (2019), The Internet Society.

[11] Tang, Y., Li, D., Li, Z., Zhang, M., Jee, K., Xiao, X., Wu, Z., Rhee, J., and andQun Li, F. X. Nodemerge: Template based efficient data reduction for big-data causality analysis. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2018), CCS '18, ACM.

[12] Xu, Z., Wu, Z., Li, Z., Jee, K., Rhee, J., Xiao, X., Xu, F., Wang, H., and Jiang, G. High fidelity data reduction for big data security dependency analyses. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2016), CCS '16, ACM, pp. 504–516.