HIT: Hybrid-mode Information Flow Tracking with Taint Semantics Extraction and Replay

Yu-Hsin Hung, Hong-Wei Li, Yu-Sung Wu, Bing-Jhong Jheng
Department of Computer Science
National Chiao Tung University
Hsinchu City, Taiwan
E-mail: hungys@hotmail.com, g6_7893000@hotmail.com, ysw@cs.nctu.edu.tw, st424204@yahoo.com.tw

Yen-Nun Huang
Research Center for Information Technology Innovation
Academia Sinica
Taipei, Taiwan
E-mail: yennunhuang@citi.sinica.edu.tw

***Abstract*—Due to increasing complexity in security attacks, it is no longer sufficient to rely on generic network-level and system-level events for attack detection. We propose the hybrid-mode information flow tracking (HIT) system to reveal application-level events by integrating static information analysis (IFA) and dynamic information flow tracking (DIFT) into the applications. Preliminary results indicate the effectiveness of the approach in detecting sensitive data leakage with a modest performance overhead.**

Keywords—decoupled dynamic information flow tracking, static information flow analysis, taint propagation, application logic vulnerabilities, online system, anomaly detection

I. INTRODUCTION

While security attacks based on common vulnerabilities have received a lot of attention, we anticipate that targeted attacks [1, 2] will soon become a major concern. This type of attacks may exploit zero-day vulnerabilities and conceal their actions by following the legitimate code paths in an application. Early detection of the attacks would require deep insights into the application logic. We propose the hybrid-mode information flow tracking (HIT) to address the challenge by offering an integrated process of static information flow analysis (IFA) and dynamic information flow tracking (DIFT). IFA helps us understand the likely flow paths of information in a program [3-6], while DIFT [7, 8] monitors the actual flows of information at runtime. The unique hybrid-mode design minimizes the runtime overhead of DIFT and offers a streamlined IFA process at the same time.

II. HYBRID-MODE INFORMATION FLOW TRACKING

HIT consists of three main steps as illustrated in Fig. 1. It minimizes the runtime overhead of dynamic information flow tracking by decoupling the tracking logic from program execution as in [9-11]. The decoupling process is based on compile-time extraction of language-independent taint semantics intermediate representation (TSIR) from the target program and offline replay of the TSIR. This is simpler in design than relying on symbolic execution as in [10, 11]. As shown in TABLE I, TSIR can be seen as a simplified version of LLVM IR, where only the semantics relevant to information flow analysis are retained. In the following, we will walk through the three main steps of hybrid-mode information flow tracking.

Taint Semantics Extraction. Based on the LLVM toolchain, the TSIR Translator extracts taint semantics from the LLVM IR of the target program at compile-time and translates them into an intermediate representation for information flow tracking (*i.e.,* the TSIR). Static information flow analysis can be directly carried out on the TSIR. Also, the TSIR Translator instruments the LLVM IR with runtime logging code. The instrumented LLVM IR will be compiled

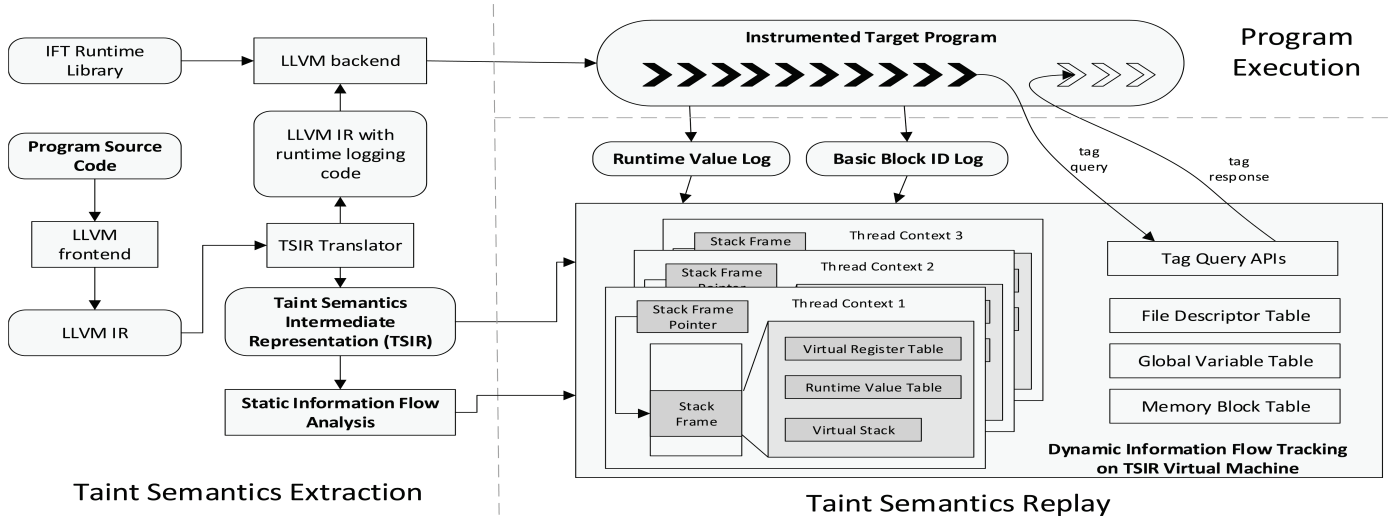

Fig. 1. Overall workflow of Hybrid-mode Information Flow Tracking

and then get linked with the IFT runtime library to produce the instrumented target program ready for dynamic information flow tracking.

Program Execution. The instrumented target program is deployed in the online environment and gets executed as usual. The instrumented runtime logging code will log the IDs of the executed basic blocks and the accompanying runtime values.

Taint Semantics Replay. DIFT is instantiated by reconstructing the control flows and the data flows of the target program based on TSIR replay with the basic block ID log and the runtime value log. Finally, HIT provides the tag query APIs for applications to synchronize program execution and DIFT when needed.

TABLE I. Examples of LLVM IR to TSIR Mapping

LLVM IR	TSIR
`%1 = alloca i32`	`MEMALLOC_STACK %1, 4`
`%2 = load i32, i32* %1`	`TAGCOPY %2, (*%1, 4)`
`%3 = add i32 %1, %2`	`TAGUNION %3, %1, %2`

III. Evaluation

We evaluated the HIT prototype on a machine equipped with Intel Core i7-3770 processor and 16GB of RAM, running Ubuntu Linux 16.04. The performance overhead of HIT is negligible for I/O-bound programs such as Unix command-line utilities (`wget`, `tar`, `md5sum`, and `sha256sum`) and server programs (`lighthttp`, `nginx`, `redis`, and `memcached`). For CPU-bound programs (`gzip` and `SPECCPU2017`), we noticed slowdowns ranging from 3x to 25x.

For effectiveness evaluation, we consider a possible misconfiguration in `lighttpd`, where a password-protected URL is mistakenly set as the homepage of the website. Clients will be able to access the sensitive HTML document without authentication by accessing the homepage URL. Fig. 2 shows the information flows induced from accessing the sensitive HTML document normally via the password-protected URL. The %6 variable in the function @connection_accept is one of the flow sources, which is due to acceptance of incoming connection. The %5 variable in the function @http_chunk_append file is the other source due to accessing the sensitive HTML document. If the sensitive document were accessed via the homepage URL, the flows circled by dashed line would not be present.

IV. Future Work and Concluding Remarks

HIT enables integrated IFA and DIFT via compile-time extraction of TSIR, lightweight execution trace logging, and offline replay of taint semantics. The approach allows it to achieve similar performance overhead as state-of-the-art decoupled DIFT systems and supports IFA at the same time. We believe the performance overhead on DIFT can be further reduced by more aggressive IFA-based optimization and extension of TSIR syntax to model the taint semantics of heavy loops in programs.

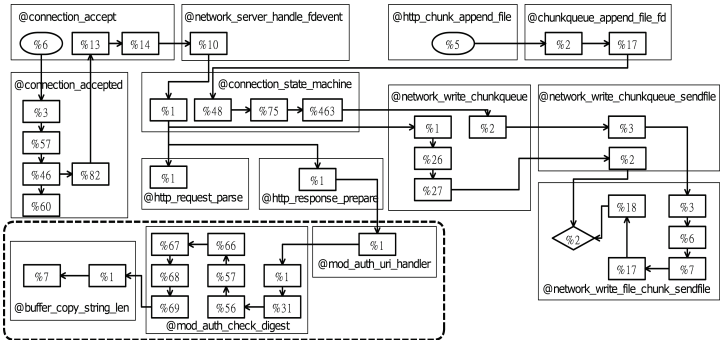

Fig. 2. Information flow of accessing the sensitive HTML document

Acknowledgment

This study is supported by the Ministry of Science and Technology of the Republic of China under grant number 104-2221-E-009-104-MY3.

References

[1] N. Virvilis and D. Gritzalis, "The big four-what we did wrong in advanced persistent threat detection?," in *Proceedings of the Eighth International Conference on Availability, Reliability, and Security (ARES)*, 2013, pp. 248-254: IEEE.

[2] L. Bilge and T. Dumitras, "Before we knew it: an empirical study of zero-day attacks in the real world," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, pp. 833-844: ACM.

[3] N. Jovanovic, C. Kruegel, and E. Kirda, "Pixy: A static analysis tool for detecting web application vulnerabilities," in *IEEE Symposium on Security and Privacy*, 2006.

[4] M. I. Gordon, D. Kim, J. H. Perkins, L. Gilham, N. Nguyen, and M. C. Rinard, "Information Flow Analysis of Android Applications in DroidSafe," in *The Network and Distributed System Security Symposium*, 2015: Internet Society.

[5] Z. Yang and M. Yang, "Leakminer: Detect information leakage on android with static taint analysis," in *Third World Congress on Software Engineering*, pp. 101-104.

[6] A. C. Myers, "JFlow: Practical mostly-static information flow control," in *Proceedings of the 26th ACM Symposium on Principles of Programming Languages*, pp. 228-241.

[7] W. Enck *et al.*, "TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones," *ACM Transactions on Computer Systems (TOCS),* vol. 32, no. 2, pp. 5-5, 2014.

[8] V. P. Kemerlis, G. Portokalidis, K. Jee, and A. D. Keromytis, "libdft: Practical dynamic data flow tracking for commodity systems," in *Proceedings of the 8th ACM Conference on Virtual Execution Environments*, pp. 121-132.

[9] K. Jee, V. P. Kemerlis, A. D. Keromytis, and G. Portokalidis, "ShadowReplica: efficient parallelization of dynamic data flow tracking," in *Proceedings of the ACM Conference on Computer & Communications Security*, 2013, pp. 235-246.

[10] J. Ming, D. Wu, J. Wang, G. Xiao, and P. Liu, "StraightTaint: Decoupled offline symbolic taint analysis," in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, 2016, pp. 308-319.

[11] J. Ming, D. Wu, G. Xiao, J. Wang, and P. Liu, "TaintPipe: Pipelined Symbolic Taint Analysis," in *USENIX Security Symposium*, 2015, pp. 65-80.