

BY SANG KIL CHA /KAIST AND ZHENKAI LIANG /  
NATIONAL UNIVERSITY OF SINGAPORE

# Asia's Surging Interest in Binary Analysis

BINARY CODE ANALYSIS (binary analysis, for short) is a vital security approach for protecting commercial off-the-shelf (COTS) software and understanding malware, where there is no source code available. From the perspective of computer security, it is imperative to analyze binary code, as source-level scrutiny does not always reveal lurking software bugs due to compiler or interpreter misbehavior.

Since the late 1990s, there has been significant research interest worldwide on binary analysis.

The BitBlaze project (by Carnegie Mellon University and University of California, Berkeley)<sup>14</sup> is one of the few pioneering research prototypes that incorporates a variety of tools for binary analysis, such as VINE for static analysis, TEMU for dynamic analysis, and Rudder for symbolic execution. Following up on BitBlaze, BAP (by Carnegie Mellon University)<sup>3</sup> provides a wealth of APIs that can be used to build a custom binary analyzer, while DECAF (by Syracuse University)<sup>7</sup> provides efficient, platform-neutral support for dynamic binary analysis. Angr (by University of California, Santa Barbara)<sup>13</sup> offers a user-friendly platform for common binary analysis tasks, such as disassembly, instrumentation, and symbolic execution that are utilized by an active user community.

In recent years, the expanding cyber infrastructure supporting governments and industry sectors in Asia has ignited strong interests in software security, leading to a surge of research activities in binary analysis in the region. Thanks to the broad international support, researchers have begun to drive new ideas and delve into various research topics in binary analysis, such as automatic exploit generation, vulnerability discovery, and automated reverse-engineering. This article outlines the key research developments and trends in binary analysis led by researchers in the East Asia and Oceania region.

## Growing Research Interest in Vulnerability Discovery and Exploit Generation

Fuzzing is a popular technique to discover software vulnerabilities by randomly or semi-randomly generating inputs to tested software programs. It has been an area of focus for researchers from China, Korea, and Singapore. AFLFast (from the National University of Singapore)<sup>2</sup> is one noteworthy project used by many security practitioners today to find security vulnerabilities in software. It served as the catalyst for grey-box fuzzing research: many



research papers on grey-box fuzzing came out after its release.<sup>12</sup> CollAFL (by Tsinghua University, China)<sup>6</sup> and Eclipser (by the Korea Advanced Institute of Science and Technology, or KAIST)<sup>4</sup> are examples of regional efforts in improving fuzzing efficiency.

Another research focus is on new types of memory vulnerabilities and exploits. Data-oriented exploitation (by National University of Singapore)<sup>8,9</sup> is a new type of memory error exploit that works by manipulating non-control data of the program without hijacking the control flow of a target program, which brings the expressiveness of data-oriented exploits to a new level.

Automatic Exploit Generation (AEG, by Carnegie Mellon University)<sup>1</sup> is a pioneering work of automatically finding and exploiting the vulnerabilities of a program, which incorporates analysis techniques such as symbolic

execution and fuzzing. AEG is important for software security, as one can apply the technique to discover vulnerabilities and quickly fix security-relevant vulnerabilities prior to the release of software products. AEG typically requires binary code, as it is not possible to figure out the exact memory layout by simply looking at the source code. The importance of AEG was realized by the Cyber Grand Challenge (CGC), the first hacking competition between machines hosted by DARPA in 2016. To expedite the AEG process, one needs to search for exploitable states in program paths diverging from crashing inputs found by fuzzing. Revery (by the University of Chinese Academy of Sciences and Tsinghua University, China)<sup>15</sup> tackles this challenge by adopting a control-flow stitching technique.

Since the CGC, several Asian countries have started to organize similar

competitions. Korea ran the AI-based Automated Vulnerability Discovery Challenge in 2018, and Japan also hosted the Automatic Cyber Hacking Challenge at the Code Blue Conference in 2018. Both competitions were focused on attacks, that is, binary-level exploitation, rather than defenses, unlike CGC, where binary patching played a crucial role. However, the Korean Ministry of Science, ICT, and Future Planning (MSIT) have announced this year's competition would include both attacks and defenses.

### Efforts in Building Scalable Binary Analysis Frameworks

Binary analysis faces a constant challenge due to the ever-increasing demands of researchers and cybersecurity responders. The dramatic increase of binary analysis tasks calls for frameworks that reduce human effort and boost productivity, which



## The expanding cyberinfrastructure supporting governments and industry sectors in Asia has ignited strong interests in software security, leading to a surge of research activities in binary analysis in the region.




can be scaled to meet real-world demands. Based on the common abstractions used in binary analysis, such as intermediate representation (IR), instruction semantics, data flow, control flow, and others, research in the region aims to develop automatic methods to generate abstractions and optimize analysis processes, as well as enabling analysis by various tools to inter-operate.

Many of the efforts carried out in the Asia region aim to produce solid building blocks for scalable binary analysis frameworks. For example, writing the specification of instructions from scratch is largely an error-prone task; instruction-set manuals typically comprise thousands of pages of descriptions written in natural language. To write a binary analysis front-end, which translates binary code into an IR, one should carefully read the manuals and implement the logic. This process requires tremendous engineering effort and thus, many researchers consider it too costly to investigate. TaintInduce (by the National University of Singapore and the Chinese Academy of Sciences)<sup>5</sup> is a project to automatically generate taint rules (or data-flow properties) without manual specifications; it infers taint rules based on observations of instruction executions. In addition, TaintInduce proposes a common definition and API for taint rules, enabling follow-up work to be built on a common knowledge base.

In 2019, researchers from KAIST made public their binary analysis framework, called B2R2.<sup>10</sup> This was the first attempt to build a binary analysis framework in this region. It focused on optimizing the performance and accuracy of a binary analysis front end, which was often neglected by binary analysis researchers, as the front end had been regarded as a simple translation module. However, they presented various optimization techniques, including parallel lifting and big-integer splitting, that achieve an order-of-magnitude improvement in the performance of the front end. The same research team published a novel technique for finding semantic bugs that appeared in IRs for binary analysis, which can

benefit any binary analysis framework available today.<sup>11</sup>

### Concluding Remarks

Binary analysis has been gaining popularity in Asia. Built on the momentum of vulnerability discovery and exploit generation, as well as the foundational work to build scalable platforms for binary analysis, we hope to see more active regional research collaboration in the field. With extended collaborative effort, we believe researchers in this region will trigger major technological breakthroughs in the future. 

### References

1. Avgerinos, T., Cha, S.K., Hao, B.L.T. and Brumley, D. AEG: Automatic exploit generation. In *Proceedings of the Network and Distributed System Security Symp.*, 2011.
2. Bohme, M., Pham, V.-T. and Roychoudhury, A. Coverage-based grey-box fuzzing as Markov chain. In *Proceedings of the ACM Conf. Computer and Communications Security*, 2016.
3. Brumley, D., Jager, I., Avgerinos, T. and Schwartz, E.J. BAP: A binary analysis platform. In *Proceedings of the Intern. Conf. Computer-Aided Verification*, 2011.
4. Choi, J., Jang, J., Han, C. and Cha, S.K. Grey-box concolic testing on binary code. In *Proceedings of the Intern. Conf. Software Engineering*, 2019.
5. Chua, Z., Wang, Y., Băluță, T., Saxena, P., Liang, Z. and Su, P. One engine to serve'em all: Inferring taint rules without architectural semantics. In *Proceedings of the Network and Distributed System Security Symp.*, 2019.
6. Gan, S. et al. CollAFL: Coverage sensitive fuzzing. In *Proceedings of the IEEE Symp. Security and Privacy*, 2018.
7. Henderson, A., Prakash, A., Yan, L.K., Hu, X., Wang, X., Zhou, R. and Yin, H. Make it work, make it right, make it fast: building a platform-neutral whole-system dynamic binary analysis platform. In *Proceedings of the Intern. Symp. Software Testing and Analysis*, 2014.
8. Hu, H., Chua, Z., Adrian, S., Saxena, P. and Liang, Z. Automatic generation of data-oriented exploits. In *Proceedings of the USENIX Security Symp.*, 2015.
9. Hu, H., Shinde, S., Adrian, S., Chua, Z., Saxena, P. and Liang, Z. Data-oriented programming: On the expressiveness of non-control data attacks. In *Proceedings of the IEEE Symp. Security and Privacy*, 2016.
10. Jung, M., Kim, S., Han, H., Choi, J. and Cha, S.K. B2R2: Building an efficient front-end for binary analysis. In *Proceedings of the NDSS Workshop on Binary Analysis Research*, 2019.
11. Kim, S., Faerevaag, M., Jung, M., Oh, S.J.D., Lee, J. and Cha, S.K. Testing intermediate representations for binary analysis. In *Proceedings of the Intern. Conf. Automated Software Engineering*, 2017.
12. Manès, V.J. et al. The art, science, and engineering of fuzzing: A survey. *IEEE Trans. Software Engineering*, 2019.
13. Shoshitaishvili, Y. et al. (State of) the art of war: Offensive techniques in binary analysis. In *Proceedings of the IEEE Symp. Security and Privacy*, 2016.
14. Song, D. et al. BitBlaze: A new approach to computer security via binary analysis. In *Proceedings of the Intern. Conf. Information Systems Security*, 2008.
15. Wang, Y., Zhang, C., Xiang, X., Zhao, Z., Li, W., Gong, X., Liu, B., Chen, K., Zou, W. Revery: From proof-of-concept to exploitable (one step towards automatic exploit generation). In *Proceedings of the ACM Conf. Computer and Communications Security*, 2018.

**Sang Kil Cha** is an assistant professor at KAIST, South Korea.

**Zhenkai Liang** is an associate professor at the National University of Singapore.

© 2020 ACM 0001-0782/20/4