FEAST'20: Fifth Workshop on Forming an Ecosystem Around Software Transformation

Kevin W. Hamlen
webmaster@marysville-ohio.com
The University of Texas at Dallas
Richardson, Texas, USA

Long Lu
l.lu@northeastern.edu
Northeastern University
Boston, Massachusetts, USA

ABSTRACT

The Fifth Workshop on Forming an Ecosystem Around Software Transformation (FEAST) provides a forum for presentation and discussion of new tools, methodologies, and techniques facilitating the automated or semi-automated transformation and analysis of software executables for improving their security and efficiency without the benefit of any original source code whence they were developed. Late-stage software customization of this form is of particular benefit to security-conscious software consumers who must use closed-source or source-free binary software components in mission-critical settings, or who must harden software against newly emerging attacks not anticipated during the software's original design and development. However, code analysis and transformation becomes much more difficult without the aid of source-level information to provide a context for its intended operation. This outstanding challenge motivates the FEAST Workshop's goal of forming a robust ecosystem of strategies and tools for accomplishing source-free binary code transformation reliably and on-demand.

CCS CONCEPTS

• **Security and privacy → Software and application security**; • **Software and its engineering → Software post-development issues**.

KEYWORDS

binary software, software debloating, software de-layering, software security hardening, binary rewriting, software transformation

ACM Reference Format:
Kevin W. Hamlen and Long Lu. 2020. FEAST'20: Fifth Workshop on Forming an Ecosystem Around Software Transformation. In *FEAST '20: Workshop on Forming an Ecosystem Around Software Transformation, November 13, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/XXXXXX.XXXXXX

1 INTRODUCTION

Software complexity and size has increased exponentially over the past several decades. For example, the Linux kernel presently consists of about 27.8 million lines of source code [1]—an increase of about 85% since 2012—and Google's codebase has been estimated to contain in the vicinity of *2 billion* lines of source code [3]. One of the root causes behind this rapid growth concerns the market forces that drive the modern software development business model, which tends to prefer feature-addition to downsizing. Software with many features is more likely to appeal to a wider variety of consumers, and is therefore easier to sell even if any individual user does not make use of a majority of the available features. Software bloat also tends to be synergistic. Code designed to be compatible with other software or that incorporates third-party modules must continue to grow with those other products in order to accommodate all their new features.

Unfortunately, software feature growth tends to have an inverse relationship to its security, reliability, and efficiency. Each new feature is a new potential point of failure and a new exploitation target for adversaries, and high-complexity software tends to have more layers of abstraction and indirection to support feature interactions. These disadvantages can be unacceptable for software consumers for whom software security, reliability, and efficiency are paramount, such as military, critical infrastructure, and healthcare consumers. Such consumers often must nevertheless resort to deploying bloated, closed-source software for practical reasons, such as its affordability, maintainability, compatibility, and availability, even if they need only a small subset of its functionalities.

Rather than fight these market forces directly, which is unlikely to be effective, the FEAST Workshop is devoted to improving the feasibility and effectiveness of *late-state software transformation*. Late-stage transformations modify low-level (usually binary) software after it has already been designed, developed, and compiled into a distributable product. Such technologies offer code consumers a facility to customize software to their particular requirements, such as by removing unneeded features, stripping out unnecessary complexity, or adding hardened security defenses against dangerous attacks. Source-free software transformation challenges of particular interest include:

- **software debloating**, which concerns the removal of software behaviors, code, or data that is unneccessary for a given consumer's needs;
- **software de-layering**, which removes levels of indirection or abstraction layers that impede efficiency;
- **software security hardening**, which concerns adding extra security checks and other defenses to code in order to thwart sophisticated attacks, such as control-flow hijacking and code-reuse attacks (e.g., [4]);
- **post-deployment patching**, which allows binary code to be more easily modified to replace or remove functionalities;

FEAST '20, November 13, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7089-9/20/11…$15.00
https://doi.org/10.1145/XXXXXX.XXXXXX

- **attack surface discovery and reduction**, which discovers and mitigates potential opportunities for abuse and compromise of binary software products;
- **software self-healing**, which transforms software to detect and remediate faults unanticipated by its authors;
- **transformation-aware reverse-engineering**, which lifts low-level software to a higher-level form amenable to analysis, and yet allows it to be lowered back to an executable form without sacrificing efficiency and compatibility; and
- **low-level formal methods**, which extend automated theorem proving, model-checking, and type-based verification typically used at the source level for high assurance code down to executable binaries.

The goal of FEAST is to cultivate a robust ecosystem of these and other technologies relevant to practical, effective customization of binary software without the aid of source code or developer support.

2 FIFTH WORKSHOP PROGRAM

The fifth FEAST workshop consists of three keynote addresses and six original research paper presentations. All submitted papers were independently reviewed by 3 members of the program committee, who scored each assigned manuscript on a scale of 1 (reject) to 5 (strong accept). Papers were accepted if a majority of reviewers favored acceptance. Of the 8 submitted works this year, this resulted in 6 acceptances, or an overall acceptance rate of 75%.

2.1 Invited Keynote Speakers

Ryan Craven is a Program Manager for the Office of Naval Research (ONR), where he manages ONR's Total Platform Cyber Protection (TPCP) program [2]. TPCP supports late-stage software customization/specialization and complexity reduction science and technology projects that offer potential for advancement and improvement of security and efficiency of Navy and Marine Corps systems and software. Supported projects include research on functionality identification and reduction, de-bloating and de-layering, addition of security constructs, verification and validation, and approaches that support and complement the above.

Alexey Loginov is the Vice President of Research at GrammaTech, a leading cyber-security research company. At GrammaTech, he oversees both technical and business aspects of the Research Division. Dr. Loginov's research work at GrammaTech has focused on binary analysis technologies, including both formal and heuristic techniques. A key contribution of his early work at GrammaTech was the addition of Binary-Analysis capabilities to the CodeSonar vulnerability-detection tool. In recent years, work in his group began to explore statistical, machine-learning, and evolutionary-computation techniques in an effort to address challenges of scalability and precision in program analysis and repair. Dr. Loginov received a Ph.D. in Computer Sciences from the University of Wisconsin, where his thesis exposed a new connection between machine learning and program analysis. Before joining GrammaTech, he worked at Hewlett-Packard and the IBM T.J. Watson Research Center.

R. Sekar is a SUNY Empire Innovation Professor and the Associate Chair of the Computer Science Department at Stony Brook University, where he directs the Secure Systems Lab. He received his Bachelor's degree in Electrical Engineering from IIT, Madras (India), and his Ph.D. in Computer Science from Stony Brook. Sekar's research interests span software and systems security. He is best known for his work on automated vulnerability mitigation, including randomization and taint-based techniques; information-flow based malware containment; intrusion detection and attack campaign investigation; and binary analysis and instrumentation. Sekar's research in these areas has been funded by several grants from AFOSR, DARPA, NSF and ONR, as well as the industry. He has supervised over 125 students, including four postdoctoral and international visiting researchers, 20+ Ph.D.s, and 80+ Master's. Sekar has received SUNY Chancellor's award for Excellence in Research, SUNY Research Foundation's Research and Scholarship award, Best paper awards at USENIX Security and Annual Computer Security Applications Conferences and honorable mention for best paper at SACMAT.

3 WORKSHOP ORGANIZATION

The 2020 FEAST Workshop's technical program committee consists of 15 members:

- Michael Brown (Georgia Institute of Technology)
- Lorenzo De Carli (Worcester Polytechnic Institute)
- Michael Franz (University of California, Irvine)
- Vasileios Kemerlis (Brown University)
- Tian Lan (George Washington University)
- Byoungyoung Lee (Seoul National University)
- Zhiqiang Lin (Ohio State University)
- Jiang Ming (University of Texas at Arlington)
- Michalis Polychronakis (Stony Brook University)
- Eric Schulte (GrammaTech)
- Nik Sultana (University of Pennsylvania)
- Guru Venkataramani (George Washington University)
- Maverick Woo (Carnegie Mellon University)
- Dinghao Wu (The Pennsylvania State University)
- Wenfei Wu (Tsinghua University)

4 ACKNOWLEDGMENTS

Is ONR providing funding this year? If so, acknowledge them here.

REFERENCES

[1] Swapnil Bhartiya. 2020. Linux in 2020: 27.8 Million Lines of Code in the Kernel, 1.3 Million in systemd. *Linux.com* (January 2020).
[2] Sukarno Mertoguno, Ryan Craven, Daniel Koller, and Matthew Mickelson. 2018. Reducing Attack Surface via Executable Transformation. In *Proceedings of the IEEE Cybersecurity Development Conference (SecDev)*. 138.
[3] Rachel Potvin. 2015. Why Google Stores Billions of Lines of Code in a Single Repository. Systems @Scale, https://youtu.be/W71BTkUbdqE.
[4] Hovav Shacham. 2007. The Geometry of Innocent Flesh on the Bone: Return-into-libc Without Function Calls (on the x86). In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*. 552–561.