Amit Seal Ami - RESEARCH STATEMENT

My research interest lies in the intersection of Security and Software Engineering, where I study the existing mechanisms of ensuring security in software, i.e., methods and techniques used by software engineers/developers for creating and maintaining software with security in mind. Particularly, during the course of my Ph.D., I focused on Static analysis-based Security Testing tools (SASTs) that offer automated security analysis for modern software and services and is a multi-billion-dollar industry to meet organizations' ever-increasing security needs. Furthermore, such techniques have gained worldwide attention because of the recent high-profile attacks and exploits across the public sector *e.g.*, SolarWind [1], triggering responses from both corporate and government entities, such as emphasizing security through the improvement of existing approaches, *e.g.*, Static Application Security Testing (SAST) tools. In essence, the use and dependence on program-analysis techniques will only increase to ensure software security.

Through the course of my work, I have designed tailored approaches for *evaluating* the SASTs to find design and implementation flaws by adapting mutation testing techniques from software engineering. Furthermore, I have applied qualitative research techniques to analyze the *perspectives and experiences* related to SASTs in practice, to identify the gaps that affect the effective application of SASTs. In other words, my work focuses on *systematically identifying the design and implementation flaws*, and the *underlying factors influencing* the *actualization* and *addressing* of these flaws. I achieve this by adapting existing techniques from multiple domains, e.g., software engineering and social sciences, to help build *secure* software. My work, therefore, focuses on both *foundational science* and *applicability*, as it contributes by adapting and introducing new techniques that can address the growing concern of security in the ever-changing landscape of software. This has resulted in multiple publications in top-tier security venues such as IEEE S&P (2022 [2], 2024 [3] - *Distinguished Paper*), and USENIX Security Symposium (2024 [4]), and in software engineering venues, namely ICSE (2021 [5]) and ESEC/FSE (2023 [6]). Furthermore, my work relies on rigorous empirical validation, collaborating with industry practitioners and researchers whenever possible. In short, my research consists of the following thrusts (T) in the context of SASTs, described next:

T1: Mutation-based Systematic Evaluation for Discovering Gaps in SASTs

SASTs are adopted to IDEs (*e.g.*, CogniCrypt for Eclipse [7], SonarQube [8]), internal test suites (Oracle testing suite [9]), and continuous integration and deployment (CI/CD) [10], to *shift-left* the detection of vulnerabilities. Thus, SASTs are instrumental in creating safe developer ecosystems [11, 12]. However, beyond manually curated benchmarks, we do not have a systematic approach for evaluating the effectiveness of SASTs. We can not entirely rely on benchmarks either since these do not represent different complexities of real-world use cases and can be incomplete or inaccurate [2, 3].

Hence, we need a reliable and evolving technique that scales to the diversity and variations of vulnerabilities while addressing the following challenges:

C1: Diversity of Vulnerabilities, as vulnerability is an umbrella term. For example, *crypto-API* misuse related vulnerabilities can be expanded to weak encryption, or vulnerable chain of trust in SSL.

C2: Strategic, Expressive Instantiations of vulnerabilities is required for effective evaluation. Using vulnerabilities, as found in the wild, would not lead to a robust analysis.

C3: Reducing Manual Effort for Scalability to reduce delay in identifying flaws in SASTs.

I proposed that it is possible to address these challenges by developing a systematic evaluation framework that contextualizes mutation testing techniques for evaluating SASTs. As a concrete example, I implemented **M**utation based **A**nalysis of **S**tatic **C**rypto-misuse detection techniques

(**MASC**, pronounced *mask*) for evaluating SASTs that detect Java/Android Crypto-API misuse (crypto-detectors), which I detailed in IEEE S&P'22 [2], FSE'23 [5] papers. I defined general, usage-based mutation operators based on the usage characteristics of crypto-APIs from Java Cryptographic Architecture (JCA) – functions that can create misuse instances (*i.e.*, mutants) of known crypto-API misuse (C1, C2). These mutants are then injected into open-source Java/Android applications at specific locations (C3) following several strategies that represent the different expertise and intentions of developers (C2).

I evaluated nine major crypto-detectors from academia (CryptoGuard, CogniCrypt with CrySL), industry (Xanitizer, Coverity, ShiftLeft Scan) and open-source communities (GitHub CodeQL, LGTM, QARK, SpotBugs with Find-Sec-Bugs) using this approach, and revealed 19 previously unknown, including design-level, flaws. Finding most of these flaws in individual detectors (45/76) is only possible because of mutated misuse instance. Furthermore, through an impact study, I demonstrated that the flaws have impacted real systems. For example, Apache Ignite (**3.5K★ in GitHub**) contains Cipher.getInstance(key.getAlgorithm()) [13], a related misuse, even after being scanned by LGTM (LGTM has been merged to GitHub Code Scan). Here, this misuse defaults to ECB mode because only the algorithm name is specified. Currently, I am researching on expanding this approach for evaluating more crypto-detectors from industry, *e.g.*, Amazon CodeGuru and SonarQube, and developing additional mutation operators.

By responsibly disclosing to and discussing about these flaws with the SASTs, it is possible to gain insights related to the factors influencing the design, testing, and industrial application related assumptions of SASTs. Examining these assumptions, both from the SAST developers' and users' perspective, is important for improving SASTs, which forms my next research thrust, **T2**.

T2: Understanding the Gaps in Industrial Application of SASTs

As we discussed the found flaws with SAST developers, we noticed that regardless of the near-identical use cases, their implicit, internal design goals and considerations vary *dramatically.* For example, some adopt a *technique-centric approach*, *i.e.*, scope of detection is tied to limitations of chosen static analysis techniques, or a *security-centric approach*, *i.e.*, the tool aims to use any static analysis techniques necessary for accomplishing security goal(s). These *implicit* design goals come with trade-offs, thus impacting the effectiveness of these tools in practice.

On the other hand, users of SASTs can have their own set of perceptions, expectations, and beliefs in SASTs, which may or may not be aligned with the implicit design goals determined and prioritized by the developers of SASTs. Without analyzing both these sets of assumptions and addressing the gaps in between, we may not be able to develop SASTs that are effective in practice, thus leading to a false sense of security.

I led a qualitative study that investigates the assumptions, expectations, beliefs, and challenges experienced by practitioners who use program-analysis-based security-assurance tools, specifically SASTs, to address this gap. As part of this, I interviewed participants working in diverse products *e.g.*, web apps, anonymity networks, safety-critical systems, and business-critical fin-tech systems. Using reflexive thematic analysis I have identified 17 key findings related to the organizational context of SASTs (*e.g.*, **Participants generally do not trust SAST benchmarks**), expectations from SAST (**nearly all participants expressed a preference for fewer false negatives**), and impact of flawed SASTs (*e.g.*, **not too concerned about unknown flaws, as expecting manual reviews to identify missed vulnerabilities**, and **participants may hesitate to report flaws**). By distilling these findings and analyzing the state-of-the-art literature, I identified prominent insights that reveal gaps in the application of SASTs (detailed in [3]) as follows.

Perceived Developer Needs vs SAST Selection/Evaluation: While all security practitioners from the study expressed strong preference for SASTs that find real vulnerabilities, they also expressed that, generally, they do not rely on effectiveness evaluation for selection. Instead, the selection factors are cost, corporate pressure, user-friendliness, recommendations, and reputation.
Unreasonable Optimism and Lack of Reliable Objective Criteria: As practitioners unreasonably assume that SASTs detect everything they claim to, they generally are not motivated to evaluate SASTs. For those who wants to evaluate SASTs, they lack the means to do so, as they considered benchmarks as biased, or not representative of real, complex vulnerabilities.
Primary Design Priority of SASTs is Misaligned with Practice: While the research community continues to show preference towards improving precision *i.e.*, decreasing false positives for SASTs in the past decade, practitioners repeatedly expressed that they want SASTs to find critical vulnerabilities even at the cost of higher number of false positives. That is, SASTs need to first be designed for finding critical vulnerabilities, with precision as an additional, desired property.
Critical Paradox in the Combined Analysis: The following conflicting assumptions forms a critical paradox: Developers (a) are *not concerned about the undocumented flaws in SASTs due to subsequent manual analysis*, and (b) emphasized that *the key reason to use SASTs is to cover knowledge gaps, and subjectivity in manual analysis.* An undesirable outcome of this is undetected vulnerabilities in code that are missed by both manual and SAST analysis. Furthermore, since practitioners are generally hesitant to report flaws, the flaws in an SAST would continue to persist and harm most software relying on that SAST, particularly because of a lack of dedicated processes related to reporting flaws. Addressing this gap forms my third thrust, P3.

T3: Understanding the Gaps in Addressing the Flaws of SASTs

Despite decades of research in creating and improving the SASTs, several critical gaps have continued to exist throughout the duration of SASTs, affecting their overall goal of ensuring software security. The key reason is that we have not looked at these gaps closely enough. For example, while working on **T1**, I found that internal design considerations of SAST developers, such as the presence and frequency of vulnerability in the wild are used for prioritizing, or even addressing flaws [2]. Similarly, users of SAST experience difficulties, such as confidentiality, red-tapes, and lack of dedicated processes for reporting flaws to SAST developers. However, by identifying these gaps, analyzing, and addressing them head-on, we have a much more realistic way forward for ensuring software security and creating a safer developer ecosystem. Therefore, I am researching on identifying and analyzing the factors that influence the addressing (or not addressing) of flaws in SASTs. By mining open-source (*e.g.*, CodeQL) and/or public-facing (*e.g.*, SonarQube) SAST issue trackers, qualitatively analyzing the contents of the issues in depth, and associating it with quantitative data, my research aims to understand the gaps in addressing the flaws of SASTs.
Research Artifacts Produced: I have made the artifacts, and associated data of my research, whenever possible as I have committed to open science, and open source software. Implementation of several frameworks from my research resulted in the creation of open source security and software evaluation frameworks, namely MUSE [6], MASC [5], and MobiCoMonkey [14].
Grant Writing Experience: My dissertation is supported by the Coastal Virginia Center for Cyber Innovation Cybersecurity *Dissertation Fellowship*, which secured funding for my Ph.D. research for two academic years (2021-2023). Furthermore, I have performed investigation and have written grant proposal for ICT Innovation Fund, Bangladesh (equivalent of *NSF Award - small*), with Dr. Kazi Sakib as the main Principal Investigator.
Research Mentorship: As the *Lead Graduate Student* of the Secure Platforms Lab (SPL) at William & Mary, I work on fostering a good research environment within the lab, and mentor

other graduate students of this lab to improve their leadership, research, critical thinking, and communication skills. In addition to this, I organize and lead the activities of the lab, specifically reviewing published, interesting research papers and status meetings.

Media and Community Outreach: I believe that scientific findings should be shared with the community, as this results in highlighting the progress made, and bringing attention to the problematic areas that need more attention. Hence, I frequently engage with the wider computer research community, policy makers, graduate students from other universities, and the public.

My work on systematically evaluating security analysis tools, and understanding the gaps that affect their effectiveness has been featured in newspapers [15, 16]. Furthermore, I have been invited as a topic expert in *Scholarly Communication* podcast (to take place in January, 2025).

Furthermore, I have shared my research with the wider community by giving invited talks at the University of Central Florida, and the George Mason University. I have also engaged in William & Mary's outreach to software developers, as well as to high school students.

I served as the COVES Policy Fellow to the Joint Commission on Technology and Science (JCOTS), Commonwealth of Virginia , where I used my security and privacy research background to analyze consumer data policies related to the Internet of Things (IoT), NextGen TVs, and accessibility tools for higher education. Furthermore, I recommended measures to strengthen consumer privacy in these areas.

Future Research

I plan to extend my research in 2 primary directions in the future:

1. Analyzing the Implicit Factors of Soundness Compromising Bugs:
Because bugs in SASTs directly compromise the security guarantees expected from software services, it is, therefore, essential to learn about the lifecycle of bugs if we want to address *comprehensively.* In other words, it is necessary to learn, analyze, and understand the implicit assumptions made by the designers of the SASTs, who are also practitioners. More specifically, it is important to learn:

- The factors in introducing a soundness-compromising bug, *e.g.*, other bug-fixing commits or new features,
- The factors that influence the decision-making process of addressing (or not addressing) such bugs,
- the evolution or change in the decision-making process in the wild by SAST developers, and
- patterns in triaging SAST bugs.

2. Effective Reporting of False Negatives from End-user:
As my work has shown, reporting false negative issues is of paramount importance when it comes to improving security analysis tools. However, the effectiveness of such reports is often compromised because of several factors, such as non-disclosure agreements or red-tapes, lack of incentives, non-straightforward approach for reproducibility, and a lack of dedicated processes for reporting false negatives. As the world adopts security analysis tools, so will it become more important to streamline the process for reporting false negative, so that the end-users can participate in improving those tools, and in turn, improve the security of their software and systems. I aim to leverage my empirical, and qualitative research analysis techniques to create novel techniques that can identify and address the existing challenges in reporting false negatives from the perspective of end-users.

References

[1] U. S. G. A. Office, “SolarWinds Cyberattack Demands Significant Federal and Private-Sector Response (infographic) — U.S. GAO,” https://www.gao.gov/blog/solarwinds-cyberattack-demands-significant-federal-and-private-sector-response-infographic.

[2] A. Ami, N. Cooper, K. Kafle, K. Moran, D. Poshyvanyk, and A. Nadkarni, “Why Crypto-detectors Fail: A Systematic Evaluation of Cryptographic Misuse Detection Techniques,” in *2022 IEEE Symposium on Security and Privacy (S&P)*. San Francisco, CA, USA: IEEE Computer Society, May 2022, pp. 397–414. [Online]. Available: https://ieeexplore.ieee.org/document/9833582

[3] A. S. Ami, K. Moran, D. Poshyvanyk, and A. Nadkarni, “”False negative - that one is going to kill you” - Understanding Industry Perspectives of Static Analysis based Security Testing,” in *Proceedings of the 2024 IEEE Symposium on Security and Privacy (S&P)*. San Francisco, CA, USA: IEEE Computer Society, May 2024, to be published in.

[4] P. Mandal, A. S. Ami, V. Olaiya, S. H. Razmjo, and A. Nadkarni, “”Belt and suspenders” or ”just red tape”?: Investigating Early Artifacts and User Perceptions of IoT App Security Certification,” in *Proceedings of the 2024 USENIX Security Symposium (USENIX)*, Aug. 2024.

[5] A. S. Ami, S. Y. Ahmed, R. M. Redoy, N. Cooper, K. Kafle, K. Moran, D. Poshyvanyk, and A. Nadkarni, “MASC: A Tool for Mutation-Based Evaluation of Static Crypto-API Misuse Detectors,” in *Proceedings of the ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE’23), Demonstration Track*, San Francisco, Dec. 2023.

[6] A. Ami, K. Kafle, K. Moran, A. Nadkarni, and D. Poshyvanyk, “Demo: Mutation-based Evaluation of Security-focused Static Analysis Tools for Android,” in *Proceedings of the 43rd IEEE/ACM International Conference on Software Engineering (ICSE’21), Formal Tool Demonstration, Virtual (originally Madrid, Spain), May 25th - 28th, 2021*, May 2021.

[7] “CogniCrypt - Secure Integration of Cryptographic Software — CogniCrypt,” https://www.eclipse.org/cognicrypt/, accessed June, 2020.

[8] “Code Quality and Security — SonarQube,” https://www.sonarqube.org/, accessed May, 2020.

[9] “Oracle - Industrial Experience of Finding Cryptographic Vulnerabilities in Large-scale Codebases,” https://labs.oracle.com/pls/apex/f?p=94065:40150:0::::P40150_PUBLICATION_ID:6629, Jul. 2020, accessed July, 2020.

[10] “CodeQL - GitHub Security Lab,” Nov. 2020, https://securitylab.github.com/tools/codeql/.

[11] C. Kern, “Developer ecosystems for software safety,” *ACM Queue*, vol. 22, p. 73–99, 2024.

[12] ——, “Secure by design at google,” Google Security Engineering, Tech. Rep., 2024.

[13] “ignite/IgniteUtils.java at apache/ignite,” https://github.com/apache/ignite/blob/1a3fd112b02133892c7c95d4be607079ffa83211/modules/core/src/main/java/org/apache/ignite/internal/util/IgniteUtils.java#L11714, Jan. 2015, accessed Jul, 2021.

[14] A. S. Ami, M. M. Hasan, M. R. Rahman, and K. Sakib, "MobiCoMonkey - Context Testing of Android Apps," in *IEEE/ACM 5th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, May 2018, pp. 76–79.

[15] A. S. Ami, "Off-the-shelf crypto-detectors give a false sense of data security – w&m news," https://news.wm.edu/2022/09/13/off-the-shelf-crypto-detectors-give-a-false-sense-of-data-security/, accessed Dec 2024.

[16] ——, "Ieee s&p 2024 distinguished paper award — computer science — arts & sciences — william & mary," https://www.wm.edu/as/computerscience/about-contactus/news/ieee-sp-2024-distinguished-paper-award.php, accessed Nov 2024.