Immerse Students in AI-Infused Cybersecurity Through Software Process

Dr. Rajendran Swamidurai, Alabama State University

Dr. Rajendran Swamidurai is a Professor and Coordinator of Computer Science at Alabama State University. He received his BE in 1992 and ME in 1998 from the University of Madras, and PhD in Computer Science and Software Engineering from Auburn University in 2009.

Dr. Uma Kannan, Alabama State University

Dr. Uma Kannan is Associate Professor of Computer Science at Alabama State University, where she has taught since 2017. She received her Ph.D. degree in cybersecurity from Auburn University in 2017.

Immerse Students in AI-Infused Cybersecurity Through Software Process

1. Introduction

Cyberspace and the Internet have become an integral part of every nation, such as cities and coastlines. They serve as the backbone for today's economy because we perform all of our daily activities, including shopping and banking, on the Internet. [1]. Due to the COVID-19 pandemic, all organizations were compelled to transition online and must now adjust to the "always-on" environment to maintain connectivity with their consumers [2].

The expansion of the Internet, the rapid growth of cyberspace, and the always-on world have all played a significant role in the remarkable increase of cyberattacks observed in recent years. In a U.S. Senate hearing in March 2013, prominent intelligence experts warned that "in the future, the cyber threat will be the paramount threat to the nation," exceeding terrorism [3, 4]. This claim was reaffirmed in a 2019 survey of 200 worldwide CEOs and 100 senior investors with assets over one billion dollars, performed by the management consultancy EY [5 ,6]. The U.S. Agency for International Development Assessment estimates that the cost of cybercrime was $8 trillion in 2023 and could escalate to $23 trillion by 2027 [6].

Cyberspace faces a multitude of threats that are continuously evolving, originating from both cybercriminals and nation-state actors. Cybercriminals employ malware, a category of harmful software that include viruses, worms, trojans, spyware, bots, rootkits, ransomware, and others, as a means for cyberattacks [7]. The motivations behind cyberattacks encompass a wide array of illicit activities, including identity theft, data theft, espionage, and the disruption of essential operations [1]. Attacks may manifest on a variety of scales, ranging from minor incidents targeting the personal information of unsuspecting individuals on their home computers to significant operations, such as the one that incapacitated the CIA (Central Intelligence Agency) website for several hours in early February 2012 [8].

Cyberattacks are increasing in frequency and severity at an alarming rate. For example, in the early cyber period, malware threats were few, and basic pre-execution rules often detected them. However, the economic incentives have led the malware authors to create numerous automated malware development toolkits, enabling inexperienced attackers to generate or tailor their own malware by merely altering existing variants [7]. These automated malware toolkits enable cyber attackers to generate numerous mutated malware samples [7] and evade detection through techniques such as instruction virtualization, packing, polymorphism, emulation, and metamorphism from the commercial malware detection software [7, 9, 10]. To combat the exponential growth of cyberthreats, an efficient, robust, and scalable detection module is required. The old tools that rely on pre-execution rules are ineffective and impractical. We require tools based on advanced protection technologies that are capable of processing vast amounts of data and delivering long-lasting defense solutions against current and future attacks. Using AI/ML techniques to automatically learn models and patterns behind such complexity and to develop solutions to keep pace with cyberthreat evolution is one of the most prevalent approaches in the literature.

In order to safeguard the nation's key infrastructures such as energy, communication, water, food, and healthcare systems, it is essential to hire qualified cybersecurity professionals.

Unfortunately, getting enough trained cybersecurity professionals is proving to be a major obstacle for both the public and commercial sectors. The size of the cybersecurity workforce shortfall increased by 19% annually, from 4 million in 2023 to 4.8 million in 2024, according to the first look at the ISC2 cybersecurity workforce survey 2024 [11].

Universities teach students how to write computer software; but that is only a small part of what is required of graduates when they enter the workforce. Industry demands a much broader perspective: that of being equipped with technical skills in identifying requirements, designing a suitable solution, implementing the solution in software, validating that the software satisfies requirements; as well as being equipped with business skills such as estimating cost, monitoring progress, measuring effectiveness, etc. Students who are inculcated with such software engineering skills are more attractive to employers that just have software-coding abilities.

This paper explains our experience and takeaways in immersing students in real-world software engineering practices using a year-long undergraduate research project development. That is, rather than simply coding the cybersecurity research projects, they engineered the cybersecurity product. Our process walked students through producing a working solution by having them use an agile process called Collaborative-Adversarial Pair (CAP) programming [12] that specifically applies cutting-edge software industry techniques at each point in the software lifecycle.

2. Need for Light-Weight Software Process

There are several software development approaches in use today and the list expands continuously. Numerous developers employ personalized approaches in the development of their software, and others choose for commercially available methodologies. Several aspects are crucial in the process of selecting a methodology, including budgetary considerations, team size, project criticality, technology employed, documentation requirements, training needs, and available tools and approaches. The conventional project approaches commonly employed by developers are widely regarded as bureaucratic or predictive in character, and have been associated with a significant number of failure projects [13, 14]. The presence of tiresome tasks can significantly impede the efficiency of the design, development, and deployment processes, resulting in a deceleration of overall progress.

A lightweight software process refers to a software development approach characterized by a limited number of rules and practices, or those that are straightforward to adhere to. The importance of addressing alterations in requirements, as well as changes in the environment or technology, is underscored by the necessity to exhibit flexibility and adaptability. In the context of lightweight software processes, it is common practice for developers to make adjustments to the process following each build or iteration. These adjustments are made in order to address any issues that may have arisen during the project, thereby establishing a continuous improvement cycle throughout the duration of the project. The subsequent points outline the primary benefits of lightweight techniques, as supported by scholarly sources [13, 14]: 1) They demonstrate a high level of adaptability, 2) Their focus is on individuals rather than procedures. They exhibit a tendency to collaborate with individuals rather than engage in oppositional behavior, 3) The utilization of dynamic checklists serves as a complement to them, and 4) The emphasis is placed on software rather than on documents.

The presence of numerous iterative cycles within lightweight techniques offers developers increased chances to thoroughly examine the project specification and adapt it to accommodate evolving business requirements. This system allows for the incorporation of additional requirements and modifications to the existing requirements list, hence enabling the adjustment of priority as deemed necessary. A further advantage of lightweight approaches is in their emphasis on generating value-added releases and mitigating architectural risk at the early stages of a project, a task that would pose challenges within the context of a heavyweight methodology.

3. Research Project Teaching/Development Process

Our project immersed students in real-world software engineering practices during their research project development. That is, rather than simply coding the cybersecurity research projects, they engineered the cybersecurity product. Our process walked students through producing a working solution by having them use an agile process called Collaborative-Adversarial Pair (CAP) programming [12] specifically to apply cutting-edge industry techniques at each point in the software lifecycle. The CAP model has been successfully used in the senior capstone project courses for the past 10 years. Moreover, the CAP method has been used as a pilot project by Neptune Technology Group Inc. in Plano, Texas, USA. The process used incrementally throughout the project development, starting with techniques that enhance writing code, and having an engineering activity, which addresses another lifecycle activity added each week. Students worked in an interactive environment in which they are instructed on new techniques and then mentored in the use of those techniques during the project development.

The CAP process employs a synchronize-and-stabilize approach to development in which features are grouped into prioritized feature sets then built in a series of software cycles, one set per cycle (Figure 1).

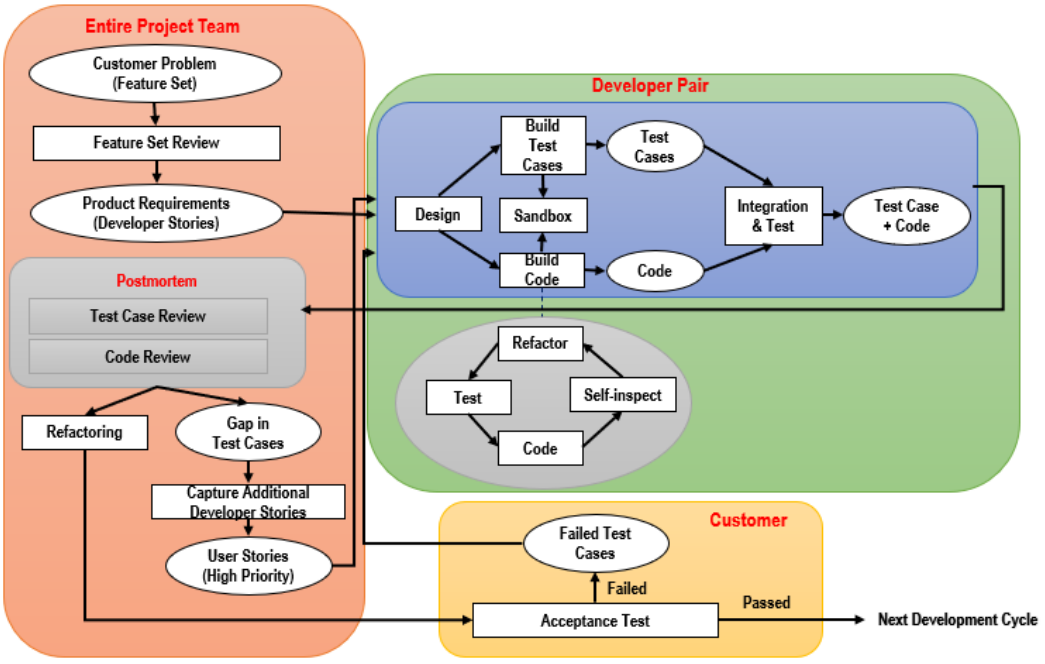

Fig.1. CAP Development Methodology

Each cycle starts with the entire project team (student pair, project mentors, and faculty advisor) reviewing the features to be built. It is here that the customer requirements (the cybersecurity problem) are translated into product requirements by converting user stories into "developer stories," which are essentially manageable units of work that map to user stories. Progress is tracked by two measures: the ratio of the number of user stories built to the total number of user stories and the ratio of the developer stories completed to the total number of developer stories to be built in the cycle. The first measure expresses progress to the customer; the second measure tracks internal progress.

After the feature review, the student-pair moves into collaborative-adversarial mode. The developers/students work together collaboratively to identify how to architect and design the features. They use this time to clarify requirements and discuss strategy. They then walk through their design with the project advisor. After the design is approved, they move into their adversarial roles. One developer/student is assigned the responsibility of implementing the design, and the other developer/student is given the task of writing black-box test cases for the various components. The goal of the implementer is to build unbreakable code; the goal of the tester is to break the code. Note that the implementers are still responsible for writing unit-level white-box tests as part of their development efforts. Once both developers have completed their tasks, they run the code against the tests. Upon discovering problems, the pair resumes their adversarial positions: the tester verifies that the test cases are valid, and the implementer repairs the code and adds a corresponding regression unit test. In some cases, the test cases are not valid and are, themselves, fixed by the tester.

At the conclusion of the test phase, the team moves to a postmortem step. Here, the team (including the project advisor) reviews the source code and the test cases. The purpose of the review is to (1) ensure the test cases are comprehensive and (2) identify portions of the code that are candidates for refactoring. The team does not walk through the code at a statement-by-statement level. This has been found to be so tedious that the participants quickly become numb to any problems. It is assumed that the majority of defects are caught in the black-box functional tests or in the white-box unit tests. Any gaps in test cases are captured as additional developer stories; refactoring tasks are done likewise. These developer stories receive a high enough priority that they are among the first tasks completed in the subsequent software development cycle.

A new development cycle begins again following the postmortem step.

4. Participants and Research Projects

4.1. Participants

This research project involved the active training of 10 undergraduate students, primarily ROTC (Reserve Officers' Training Corps) cadets from various universities within the state. Students were organized into five training groups or pairs. A pair of REU students worked on a project utilizing the CAP software development approach, under the close guidance of a project leader and in regular contact with one or more additional project leaders.

4.2. Research Projects

Though research in intrusion detection has been around for several years, applications are always changing and morphing. Current intrusion detection processes suffer from several limitations when focusing on highly vulnerable network intrusions. First, with the increasing volume of network traffic -- existing intrusion detection processes fail to analyze the vulnerabilities in time to predict possible network intrusion(s) from the chain of actions of an intruder. Second, current intrusion detection systems produce a high volume of false positive alerts. And third, current approaches consider every sequence of network vulnerability to predict future intrusions rather than analyzing the comparatively significant sequences. Instead of teaching theoretical cybersecurity concepts it will be beneficial to allow the student(s) to be involved in the design and improvements to current and next generation time series intrusion detection systems.

Network intrusion refers to unauthorized activity on closed digital networks. Network intrusions have become very common in the digital age and can be highly disruptive to the operations of any organization. Some of the popular Network intrusion attach vectors are flooding (e.g. Buffer overflow flow, DDoS), multi-routing or asymmetric routing (more than one route to the targeted network device), protocol specific attacks, trojan/malware and CGI scripts. Once an intruder is inside the Network, they can imperil the network and data security of the organization. Multiple Intrusion Prevention (IPS) and Intrusion Detection Systems (IDS) are available on the market. Machine Learning (ML) and Artificial Intelligence (AI) models can be trained to detect and prevent network intrusion and can be the first level of detection/prevention supporting the security personnel responsible for the network security of an organization [15]. Many researchers have published surveys and studies on the efficacy of using AI+ML for such network anomaly detection [16-18]. Through this research our plan was to obtain the answer for the question how we can use AI+ML to train, predict, and prevent network intrusion as a first layer of network security?

Time Series Intrusion Detection: This research topic involved developing time series intrusion detection systems that can determine highly vulnerable intrusions from network traffic. Though research in intrusion detection has been around for several years, applications are always changing and morphing (for example, the advent of cloud related services). Current intrusion detection processes suffer from several limitations when focusing on highly vulnerable network intrusions. First, with the increasing volume of network traffic -- existing intrusion detection processes fail to analyze the vulnerabilities in time to predict possible network intrusion(s) from the chain of actions of an intruder. Second, current intrusion detection systems produce a high volume of false positive alerts. And third, current approaches consider every sequence of network vulnerability to predict future intrusions rather than analyzing the comparatively significant sequences. The student(s) involved in the design and improvements to current and next generation time series intrusion detection systems. Student(s) leveraged currently developed intrusion detection processes to gather knowledge on existing vulnerabilities and their priority in terms of how they might affect the resources. [19-22]

Predicting Network Intrusions with AI+ML: As is common with all AI+ML problems the requirement of the requisite amount and quality of the data is paramount to a successful outcome. Large datasets for developing ML and AI inferencing applications are publicly available. Most of

the datasets are well defined and well curated. Some of the examples are CTU-13 (Czech Technical University) dataset – which is one of the largest and more labeled existing datasets in the Cyber Security field for botnet detection, ADFA Linux datasets - consists of Linux-based system call traces of the normal and attack types and the UNB-CIC dataset –which is published by University of New Brunswick (UNB) related to Cyber Security almost every year, collaborating with the Canadian Institute for Cybersecurity. This dataset has been categorized based on the type of attack. Two training project teams worked with these publicly available datasets, one for intrusion detection which has traditionally been based on patterns of known attacks, but with modern deployments to include other approaches for anomaly detection, threat detection and classification based on machine learning principles such as LSTM/RNN classifiers [23] and the second team for malware analysis and spam and phishing detection. We used Fully Connected Feed Forward Neural Network (FNN) and Convolutional Neural Networks (CNN) for malware analysis and K-Nearest neighbor (KNN) and shallow neural networks (SNN) for spam/phishing detection.

5. Summary and Conclusion

Cyberattacks, ranging from identity theft to life-threatening threats, pose an increasing danger to our country. To hire enough qualified security personnel, the industry faces significant obstacles. One of the difficulties in addressing cyber workforce issues is the well-documented lack of STEM graduates who are qualified to work in the cyber field. Even though STEM careers in academia and industry increasingly require technical skills for dealing with cybersecurity, undergraduate computer science courses fail to provide students with the necessary training in cybersecurity areas that integrate theory and practice. Students' employability will be significantly enhanced if they possess such skills. The overall objective of this study was to promote discovery-based learning as opposed to passive listening. This was accomplished using an agile software engineering methodology called Collaborative-Adversarial Pair programming. Our process walked students through producing a working solution for real-world cybersecurity problems.

Through this project ROTC and URM students received an entire year of training from R1 research schools and an AI/cybersecurity startup. We established five intriguing and demanding research projects in cybersecurity. These were significant and hard enough to be of interest as undergraduate Cyber Security research projects, but participants with a good background in fundamental mathematics and an introduction to computer programming were able to comprehend the projects. The projects are Security in Vehicular Ad-Hoc Networks (VANETs), Time Series Intrusion Detection, Host Layers Cyber Security Modeling, Predicting Network Intrusions with AI+ML, and Privacy/Trust and Access Control in IoT Network.

Acknowledgements

This research was funded by the Office of Naval Research (ONR) grant number N00014-21-1-2553. We would like to express our gratitude to Mr. Anthony Thomas, CIO, AnalyticalAI for his time, students’ advice, and support.

References

1. Jeh C. Johnson, Let's pass cybersecurity legislation, http://thehill.com/opinion/op-ed/217151-lets-passcybersecurity-legislation
2. "2 stores, 100M hacks. Where's cybersecurity? Our view," The Editorial Board, 7:42 p.m. EDT September 14, 2014, http://www.usatoday.com/story/opinion/2014/09/14/home-depot-target-data-breach-credit-card-editorialsdebates/15642867/?utm_source=feedblitz&utm_medium=FeedBlitzRss&utm_campaign=news-opinion
3. Cyber Security and Network Reliability, https://www.fcc.gov/encyclopedia/cyber-security-and-networkreliability
4. Cyber Security Primer, http://www.umuc.edu/cybersecurity/about/cybersecurity-basics.cfm
5. Chloe Taylor, "Cybersecurity is the biggest threat to the world economy over the next decade, CEOs say," https://www.cnbc.com/2019/07/09/cybersecurity-biggest-threat-to-world-economy-ceos-say.html, Published Jul 9 2019, Updated Mar 13, 2020.
6. KC Nwakalor, "Cybersecurity: ECONOMIC GROWTH AND TRADE (EGAT)," USAID / Digital Development, https://www.usaid.gov/sites/default/files/202310/Cybersecurity%20Briefer_Economic%20Growth.pdf
7. Yanfang Ye, Tao Li, Donald Adjeroh, and S. Sitharama Iyengar. 2017. A survey on malware detection using data mining techniques. ACM Comput. Surv. 50, 3, Article 41 (June 2017), 40 pages. DOI: http://dx.doi.org/10.1145/3073559
8. Cyber Security and Network Reliability, https://www.fcc.gov/encyclopedia/cyber-security-and-network-reliability
9. Philippe Beaucamps and ric Filiol. 2007. On the possibility of practically obfuscating programs towards a unified perspective of code protection. Journal in Computer Virology 3, 1 (2007), 3–21.
10. Eric Filiol, Gregoire Jacob, and Mickael Le Liard. 2007. Evaluation methodology and theoretical model for antiviral behavioural detection strategies. Journal in Computer Virology 3, 1 (2007), 23–37.
11. "Employers Must Act as Cybersecurity Workforce Growth Stalls and Skills Gaps Widen," ISC2 Research 2024 Cybersecurity Workforce Study First Look, https://www.isc2.org/Insights/2024/09/Employers-Must-Act-Cybersecurity-Workforce-Growth-Stalls-as-Skills-Gaps-Widen
12. Swamidurai, Rajendran, Umphress, David A., Collaborative-Adversarial Pair Programming, International Scholarly Research Notices, 2012, 516184, 11 pages, 2012. https://doi.org/10.5402/2012/516184
13. Jason P. Charvat, "Heavyweight vs. lightweight methodologies," December 03, 2002, Builder.com, https://www.techrepublic.com/article/heavyweight-vs-lightweight-methodologies-key-strategies-for-development/
14. Prabhu Selvaraj, NEED A NERD – Adapting PCSE (Practitioner Centered Software Engineering) to develop a Web Application, PhD Dissertation, Auburn University.

15. Shanbhogue, RD, Beena, BM. Survey of data mining (DM) and machine learning (ML) methods on cyber security. Ind J Sci Technol 2017; 10: 1153–1176.
16. Kwon, D, Kim, H, Kim, J, et al. A survey of deep learning-based network anomaly detection. Cluster Comput 2017; 22: 949–961.
17. Ye, Y, Li, T, Adjeroh, D, et al. A survey on malware detection using data mining techniques. ACM Comput Surv 2017; 50: 41:1–41:40.
18. Barriga, J, Yoo, SG. Malware detection and evasion with machine learning techniques: a survey. Int J Appl Eng Res 2017; 12: 7207–7214.
19. T. Atkison, S. Ponomarev, R. Smith, and B. Chen, “Feature Extraction Optimization for Network Intrusion Detection in Control System Networks.” International Journal of Network Security, vol. 20, no. 5, 2018, pp. 853–86.
20. S. Haque and T. Atkison, “An Evolutionary Approach of Attack Graph to Attack Tree Conversion,” International Journal of Computer Network and Information Security, vol. 10, no. 11, 2017, pp. 1-16.
21. S. Haque, M. Keffeler, and T. Atkison, “An Evolutionary Approach of Attack Graphs and Attack Trees: A Survey of Attack Modeling,” in Proceedings of the International Conference on Security and Management (SAM), July 2017, pp. 224–229.
22. S. Haque and T. Atkison, "A Forensic Enabled Data Provenance Model for Public Cloud,” Journal of Digital Forensics, Security and Law, vol. 13, no. 3, 2018, article 7.
23. J. Kim, J. Kim, H. L. T. Thu, and H. Kim, “Long Short Term Memory Recurrent Neural Network Classifier for Intrusion Detection,” in IEEE International Conference on Platform Technology and Service (PlatCon), 2016.