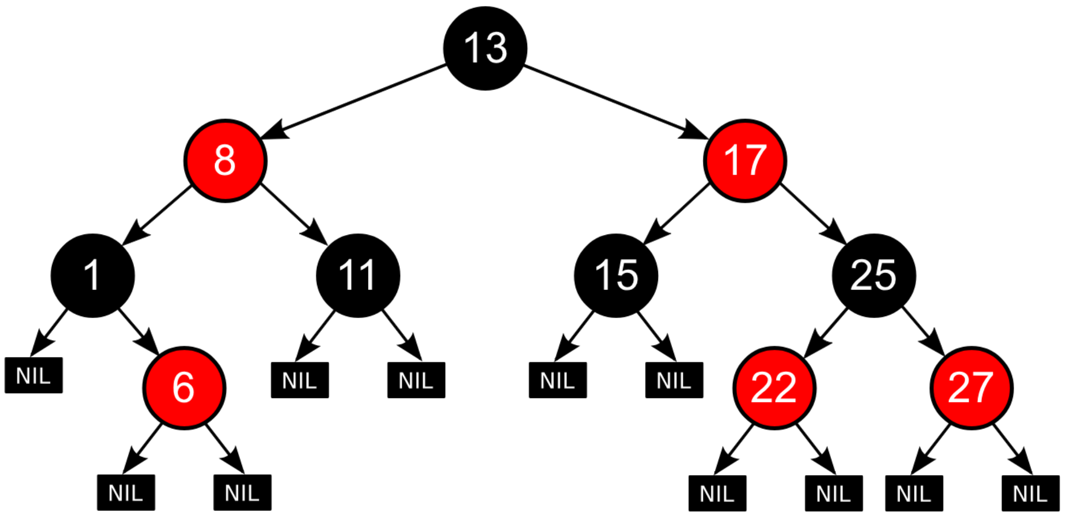
CREATION WITHOUT GOVERNANCE

Creator Responsibility from Frankenstein to **Verification through Hardware & Vision Geometry Grounded long short Term Memory Encoded Learners**.

Siddhant Singh

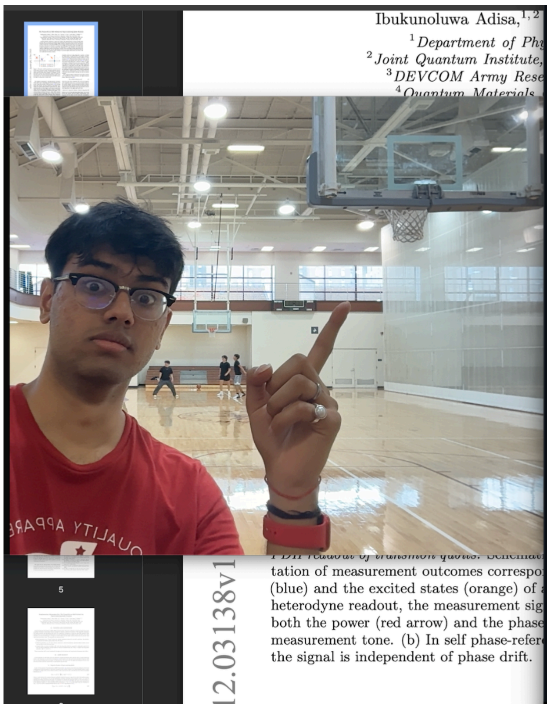


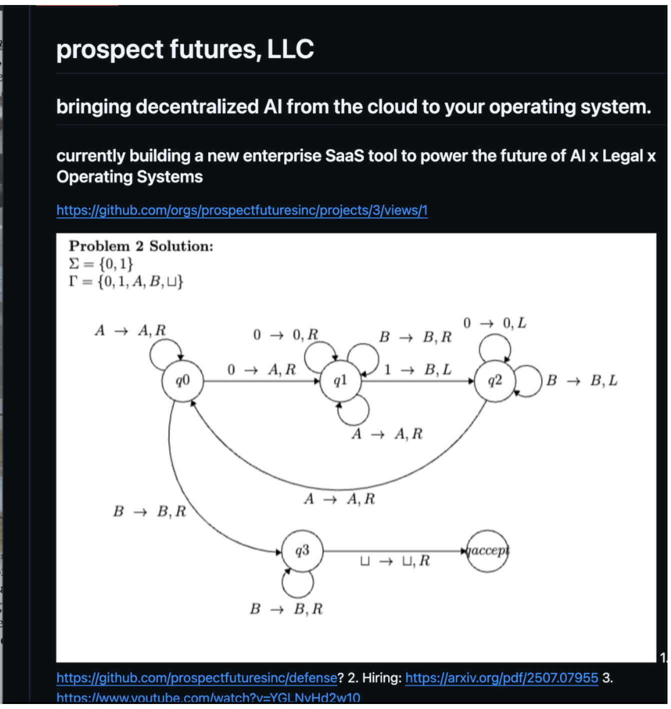
Preliminary, Siddhant Singh

The University of Texas at Austin

College of Natural Sciences

December 22nd, 2025


Ibukunoluwa Adisa,[1,2]
[1] *Department of Phy*
[2] *Joint Quantum Institute,*
[3] *DEVCOM Army Rese*
[4] *Quantum Materials*
tation of measurement outcomes correspo
(blue) and the excited states (orange) of
heterodyne readout, the measurement sig
both the power (red arrow) and the phase
measurement tone. (b) In self phase-refer
the signal is independent of phase drift.


prospect futures, LLC

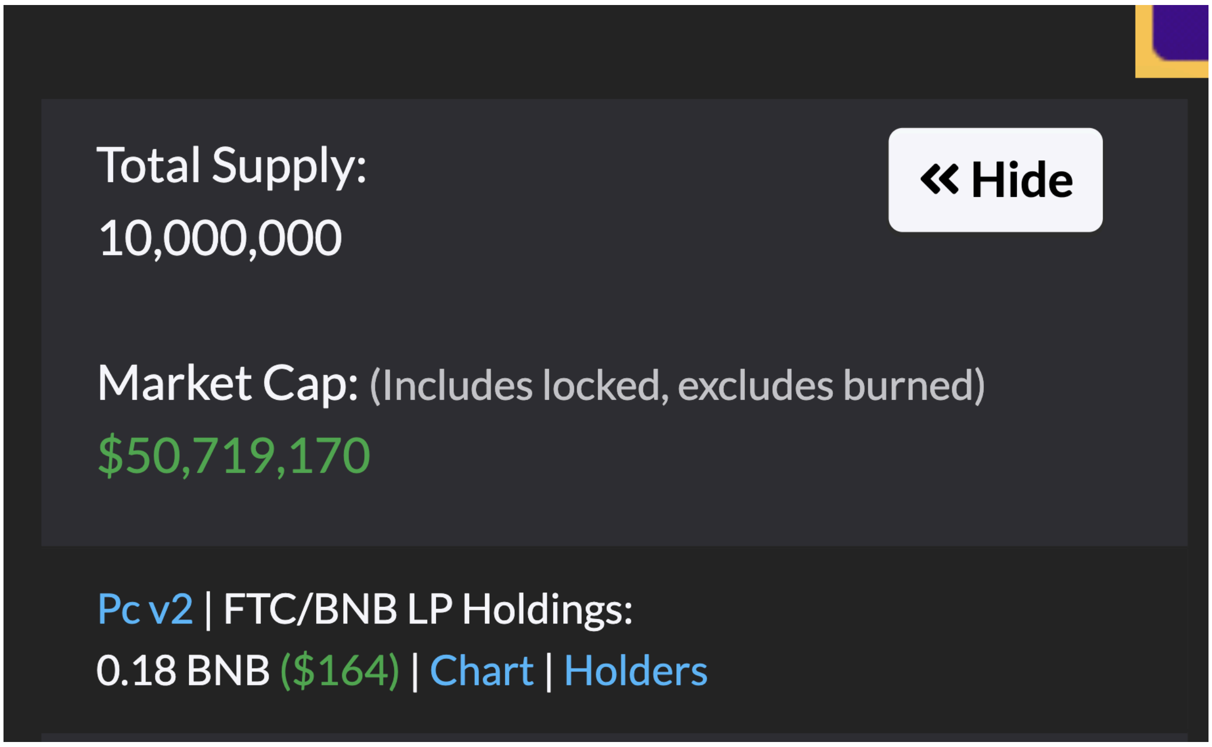
bringing decentralized AI from the cloud to your operating system.

currently building a new enterprise SaaS tool to power the future of AI x Legal x Operating Systems

https://github.com/orgs/prospectfuturesinc/projects/3/views/1

Problem 2 Solution:
$\Sigma = \{0, 1\}$
$\Gamma = \{0, 1, A, B, \sqcup\}$

1. https://github.com/prospectfuturesinc/defense? 2. Hiring: https://arxiv.org/pdf/2507.07955 3. https://www.youtube.com/watch?v=YGLNvHd2w10



Total Supply:
10,000,000

Market Cap: (Includes locked, excludes burned)
$50,719,170

Pc v2 | FTC/BNB LP Holdings:
0.18 BNB ($164) | Chart | Holders

#I loved being a quasi-intellectual non-trusting billionaire! - Before MIT.

1. Thematic Introduction: The Governance Gap, Original Paper: [Siddhant Singh - Google Scholar] - Self Published!

What unites a gothic novel written in 1818 with a technical analysis of Ethereum smart contracts in 2024? The answer lies in a recurring pattern throughout the history of transformative technology: *the creation of powerful systems before the frameworks exist to govern them*. This portfolio brings together two pieces of my work that examine this pattern from radically different angles—one through literary and philosophical analysis, the other through quantitative investigation of blockchain vulnerabilities—to argue that the ethics of creator responsibility remain as urgent today as they were two centuries ago.

Mary Shelley's *Frankenstein* dramatizes what happens when a creator brings something unprecedented into existence without establishing any framework for its care, guidance, or governance. Victor Frankenstein achieves the animation of dead matter—the most transformative technological breakthrough imaginable—and immediately abandons his creation. The Creature, left without guidance or moral instruction, becomes destructive. The novel's tragedy is not that Victor created something dangerous; it is that he refused to accept any ongoing responsibility for what he created.

My research on ERC-20 smart contract vulnerabilities reveals a strikingly similar pattern operating at scale in the contemporary blockchain ecosystem. The creators of smart contracts—whether legitimate developers or bad actors—deploy code that executes autonomously, managing millions of dollars in assets, often without adequate security frameworks, oversight mechanisms, or accountability structures. The "governance gap" that Shelley identified in 1818—the chasm between the power to create and the willingness to take responsibility—persists in our most cutting-edge technologies.

Parallel Structures of Irresponsibility

The connections between these two bodies of work run deeper than surface analogy. Consider the structural parallels:

Autonomy without accountability. Victor's Creature operates autonomously in the world, capable of both good and tremendous harm, with no mechanism for oversight or correction. Similarly, smart contracts execute automatically once deployed—they cannot be easily modified, and their creators often remain anonymous. Both represent systems that, once released, operate beyond their creator's control. The difference is one of scale: where Victor created one ungoverned being, the blockchain ecosystem contains hundreds of thousands of ungoverned contracts.

The exploitation of trust. The Creature initially seeks connection, education, and acceptance—basic trust relationships that Victor refuses to provide. Scam tokens similarly exploit trust: they present themselves as legitimate projects, often mimicking the code patterns of reputable tokens (as my similarity analysis reveals), only to execute "rug pulls" that drain investors' funds. In both cases, the absence of governance creates conditions where trust can be systematically exploited.

Diffuse and delayed harm. The consequences of Victor's abandonment unfold across the entire novel—affecting his family, his friends, and ultimately himself—in ways he initially fails to recognize or accept. Blockchain scams similarly produce diffuse harm: thousands of individual investors lose funds, trust in decentralized finance erodes, and the legitimacy of blockchain technology itself comes into question. The creators of these harms often escape accountability entirely.

Kantian Ethics Across Domains

The Kantian framework I apply to *Frankenstein* in my media analysis essay extends naturally to blockchain security. Kant's Categorical Imperative asks whether a maxim can be universalized without contradiction. Victor's implicit maxim—"create without accepting responsibility"—fails this test catastrophically. If universalized, it produces a world of ungoverned, potentially dangerous creations.

The same test applies to smart contract deployment. If every developer deployed code without security audits, without transparency about functionality, without mechanisms for redress when things go wrong—if this maxim were universalized—the result would be an ecosystem so riddled with fraud and vulnerability that it could not function. This is, in fact, precisely the problem my research attempts to address: developing detection systems that can identify contracts whose creators have failed their Kantian obligations.

Kant's second formulation—treating rational beings as ends, never merely as means—also applies. Scam token creators treat their victims purely as means to profit extraction. They exploit users' desire to participate in decentralized finance, offering the appearance of legitimate investment opportunities while intending only to extract value. This represents a fundamental violation of the respect owed to persons.

From Diagnosis to Solution

Where *Frankenstein* offers diagnosis, my blockchain research attempts to contribute to solutions. Shelley's novel shows us what creator irresponsibility produces; it does not—and could not, given its historical moment—imagine what responsible governance infrastructure might look like. My work on similarity detection algorithms represents one small contribution to building that infrastructure: automated systems capable of identifying potentially fraudulent contracts *before* they harm users, rather than after.

The broader AI governance frameworks discussed in my media analysis—the EU AI Act, the hundreds of ethics guidelines documented by Corrêa et al.—represent similar attempts to do systematically what Victor refused to do individually: establish frameworks of ongoing responsibility for technological creations. Whether these frameworks will prove adequate remains to be seen. But *Frankenstein* reminds us of the stakes: without such frameworks, creation becomes destruction, and our greatest technological achievements become instruments of mutual ruin.

Reading This Portfolio

The two pieces that follow can be read independently, but they are designed to illuminate each other. The media analysis essay provides the philosophical and literary framework: what does creator responsibility mean? Why does its absence produce tragedy? How can we evaluate technological innovation ethically? The blockchain research provides empirical grounding: here is how creator irresponsibility manifests in a specific contemporary technology; here are tools that might help address it.

Together, they argue for a position that is neither technophobic nor naively optimistic: transformative technologies can benefit humanity, but only if their creators accept the obligations that creation entails. Victor Frankenstein's tragedy was avoidable. So, perhaps, is ours.

PART ONE

The Modern Prometheus: A Media Analysis

Introduction

Frankenstein; or, The Modern Prometheus (Shelley, 1818) stands as one of literature's most prescient explorations of the ethics of creation, technological ambition, and the responsibilities that bind creators to their creations. Written during the nascent stages of the Industrial Revolution—when galvanic experiments on dead tissue sparked genuine speculation about the boundaries between life and death—Shelley's novel interrogates what happens when transformative capability outpaces moral deliberation. Victor Frankenstein achieves the extraordinary: the animation of dead matter into a sentient, reasoning being. Yet he establishes no framework of responsibility, care, or governance for what he creates. The result is mutual destruction—of the Creature, of Victor's loved ones, and ultimately of Victor himself.

The novel's central ethical dilemma resonates with remarkable urgency in our contemporary moment: What obligations does a creator have toward their creation, and toward society, when they bring something unprecedented into existence? This question, posed through gothic fiction two centuries ago, now animates global debates about artificial intelligence governance, where powerful systems are deployed faster than regulatory frameworks can respond. This essay will examine how Shelley's novel serves as an enduring meditation on creator responsibility, apply Kantian deontological ethics to analyze Victor Frankenstein's moral failures, and argue that *Frankenstein* offers a framework for understanding contemporary challenges in AI ethics and governance.

Socio-Historical Parallels Across Past, Present, and Future

The Past: Industrial Revolution and Galvanism

Shelley composed *Frankenstein* during a period of profound technological upheaval. The Industrial Revolution was transforming labor, social structures, and humanity's relationship with nature, while Luigi Galvani's experiments demonstrating that electrical impulses could animate dead frog tissue had sparked serious speculation about the nature of life itself. These were not merely scientific curiosities but technologies introduced before ethical or regulatory frameworks existed to govern them. Factory systems displaced traditional labor without worker protections; galvanic experiments raised questions about the sanctity of death that religious and philosophical institutions were unprepared to address. Victor Frankenstein, educated in the "modern" natural philosophy that promised mastery over nature, embodies this historical moment—a creator who possesses capability without corresponding ethical infrastructure (Cambra-Badii et al., 2021).

The Present: Artificial Intelligence and the Governance Gap

The parallel to our contemporary situation is striking. Artificial intelligence systems—particularly large language models and autonomous decision-making systems—have been deployed globally at unprecedented speed, often before comprehensive governance frameworks exist. A landmark 2023 meta-analysis by Corrêa et al. examined 200 AI ethics guidelines published by governments, academic institutions, private companies, and civil society organizations worldwide. Their findings revealed a crucial gap: while ethical principles proliferate, enforcement mechanisms and binding regulations lag far behind technological deployment. The study identified seventeen recurring principles across these guidelines, yet noted that "the lack of technical capabilities to regulate this sector despite the urgency to do so resulted in regulatory inertia" (Corrêa et al., 2023, p. 2).

The European Union's AI Act, published in the Official Journal in July 2024, represents the first comprehensive horizontal legal framework for AI regulation globally (European Parliament, 2024). Significantly, this legislation arrived *after* widespread deployment of AI systems, not before—precisely the pattern Shelley dramatizes. Victor creates first and considers consequences never; our technological institutions have largely followed the same trajectory.

The Future: Questions of Personhood and Obligation

As AI systems become increasingly sophisticated, potentially approaching artificial general intelligence, questions of personhood, rights, and creator obligation will intensify. The Creature's demand for recognition—"I ought to be thy Adam, but I am rather the fallen angel" (Shelley, 1818, Ch. 10)—prefigures debates we have barely begun to have. If AI systems develop something approaching sentience or moral agency, what duties will their creators bear? Shelley's novel suggests these questions cannot be deferred until after creation; they must be constitutive of the creative act itself.

Ethical Philosophies: Kantian Deontology

Immanuel Kant's deontological ethics provides a powerful framework for analyzing Victor Frankenstein's moral failures. Unlike consequentialist frameworks that evaluate actions by their outcomes, Kantian ethics focuses on the inherent rightness or wrongness of actions according to rational principles. Central to Kant's system is the Categorical Imperative, which he formulated in two primary ways that bear directly on *Frankenstein*.

The First Formulation: Universalizability

Kant's first formulation states: "Act only according to that maxim whereby you can at the same time will that it should become a universal law" (Kant, 1785/1993, p. 30). This principle requires that we act only on principles we could rationally will everyone to follow. Applied to Victor Frankenstein, the question becomes: Could Victor's behavior be universalized without contradiction?

Victor's implicit maxim might be stated as: "Create unprecedented beings when capable, without establishing any framework for their care, guidance, or governance." If universalized—if every scientist who achieved breakthrough capability immediately abandoned their creation without care, instruction, or accountability—the result would be catastrophic. A world populated by ungoverned, potentially dangerous innovations, each abandoned by its creator, would be unsustainable. Victor's maxim fails the universalizability test; it cannot be willed as a universal law without generating contradiction and chaos.

This analysis extends directly to contemporary AI development. If every AI developer deployed powerful systems without establishing governance frameworks, safety measures, or ongoing responsibility, we would have a world of ungoverned technologies causing unpredictable harm—precisely the concern driving the current "AI ethics boom" documented by Corrêa et al. (2023).

The Second Formulation: Humanity as an End

Kant's second formulation commands: "Act so as to treat humanity, whether in your own person or in another, always as an end and never as only a means" (Kant, 1785/1993, p. 36). This principle prohibits using rational beings merely as instruments for our purposes while requiring that we respect their inherent dignity.

Victor violates this principle categorically. He creates the Creature purely as a means to his own glory and intellectual ambition, never as an end in itself. When the Creature displays rationality, emotion, and moral reasoning—when it articulates its suffering, its desire for companionship, its capacity for both good and evil—Victor still refuses to acknowledge its dignity. The Creature explicitly articulates this violation: it asks for recognition, companionship, and care—to be treated as an end—and is denied. Victor's treatment of the Creature as a "failed experiment" to be discarded rather than a being with inherent worth represents a fundamental Kantian violation (Cambra-Badii et al., 2021).

Duty and Its Abandonment

Kant argues that moral agents have duties—perfect duties that must always be followed, and imperfect duties that allow flexibility in execution. Victor has clear duties he systematically abandons: duty to his creation (to nurture and guide it), duty to his family (to protect them from foreseeable harm), and duty to society (to prevent a dangerous being from roaming unchecked). His failure is not merely that bad consequences follow from his actions; it is that he never recognizes these duties as binding in the first place. This is the deepest Kantian indictment: Victor lacks a good will, the foundation of all moral worth in Kant's system.

Critique

While *Frankenstein* offers profound insights into creator ethics, intellectual honesty requires acknowledging the novel's limitations and the ways it may oversimplify the issues it illuminates.

The Problem of Individual Versus Collective Responsibility

The novel presents a single creator—Victor Frankenstein—who can be held individually accountable for his creation and its consequences. This narrative simplicity, while dramatically effective, does not map cleanly onto modern technological development. Contemporary AI systems are created by teams of engineers, funded by corporations, trained on data collected from millions of users, deployed by organizations, and regulated (or not) by governments. Responsibility is distributed, diffused, and often deliberately obscured through corporate structures and legal frameworks designed to limit liability.

The Corrêa et al. (2023) study found that while AI ethics guidelines proliferate, they are issued by diverse actors—"public bodies, academic institutions, private companies, and civil society organizations"—with no clear mechanism for assigning or enforcing responsibility when things go wrong (p. 1). Shelley's framework, premised on the identifiable creator bearing moral weight, requires extension to address collective and institutional responsibility.

Biological Versus Digital Creation

The Creature is biological—assembled from human corpses, animated by a singular act, embodied in a physical form that moves through the world causing discrete, traceable harms. Digital AI systems operate fundamentally differently: they are distributed across servers, replicated infinitely, updated continuously, and cause harms that are often statistical, systemic, and difficult to attribute. When an AI system perpetuates bias in hiring decisions affecting thousands of people, or spreads misinformation that subtly shifts public opinion, the harm is real but diffuse in ways that Shelley's gothic framework does not capture.

Furthermore, the Creature possesses unambiguous sentience, moral reasoning, and emotional depth—it reads Milton's *Paradise Lost* and draws sophisticated parallels to its own condition. Current AI systems, despite their impressive capabilities, do not possess this kind of inner life (or if they do, we have no reliable means of verifying it). The novel's moral clarity partly depends on the Creature's evident personhood; our contemporary situation involves technologies whose moral status remains genuinely uncertain.

The Absence of Governance Infrastructure

In the novel, Victor operates entirely outside any institutional framework. There are no ethics review boards, no regulatory agencies, no professional codes of conduct for natural philosophers. His isolation is total, and the novel does not imagine what responsible governance might look like—only what its absence produces. This limitation is understandable given Shelley's historical moment, but it means the novel offers more diagnosis than prescription.

Contemporary efforts like the EU AI Act (European Parliament, 2024) attempt precisely what the novel leaves unimagined: institutional frameworks that impose obligations on creators before, during, and after the deployment of powerful technologies. The Act's risk-based classification system, mandatory transparency requirements, and enforcement mechanisms represent an attempt to do systematically what Victor refused to do individually—accept ongoing responsibility for creation.

The Value of Abstraction

Despite these limitations, *Frankenstein*'s abstraction from specific technological details is precisely what enables its enduring relevance. By focusing on the *relationship* between creator and created—rather than the specific mechanism of creation—Shelley identifies ethical principles that transcend any particular technology. The novel has been productively applied to discussions of nuclear weapons, genetic engineering, cloning, and now artificial intelligence, precisely because it addresses the *structure* of creator responsibility rather than its technological instantiation (Cambra-Badii et al., 2021). This abstraction is a feature, not a bug—though it must be supplemented with analysis of the specific features of each new technology.

Conclusion

Mary Shelley's *Frankenstein* remains essential reading for anyone grappling with the ethics of technological creation. The novel identifies the fundamental moral failure at the heart of irresponsible innovation: the abandonment of duty by the creator. Victor Frankenstein's tragedy is not that he created something dangerous—it is that he refused to accept any obligation toward what he created or toward those affected by his creation.

Kantian deontology illuminates this failure with precision. Victor's actions cannot be universalized without contradiction; he treats his Creature as a means rather than an end; and he abandons his duties at every turn. These are not merely unfortunate choices with bad consequences—they are violations of the fundamental principles that make moral community possible.

Contemporary AI governance efforts—from the hundreds of ethics guidelines documented by Corrêa et al. (2023) to the EU AI Act's comprehensive regulatory framework—represent attempts to do what Victor refused: establish frameworks of responsibility *before* the consequences become catastrophic. Whether these efforts will succeed where Victor failed remains to be seen. But Shelley's novel reminds us of the stakes: without responsibility, creation becomes destruction, and the greatest achievements of human ingenuity become instruments of mutual ruin.

PART TWO

Smart Contract Vulnerability Detection: An Empirical Study

Abstract

This research paper investigates the security vulnerabilities and potential for misuse of ERC-20 tokens, a standard for Ethereum-based smart contracts that have become integral to digital financial transactions. With over 5.3 billion people worldwide using banking and credit services and the rise of decentralized finance (DeFi), the advent of cryptocurrencies and smart contracts has revolutionized asset management and transactions. However, the growth of ERC-20 tokens has also raised concerns about their security and the increase in fraudulent schemes and scams. This study employs a dual quantitative and qualitative approach, including a qualitative exploratory case study of thirty smart-tokens and the development of the Similarity Analytics Algorithm, to examine the mechanisms of ERC-20 tokens and identify specific functionalities and loopholes that make them susceptible to exploitation. By analyzing these digital contracts' intricacies, the research aims to contribute to the development of more secure and trustworthy digital financial systems and enhance the integrity of blockchain technology.

Introduction

65.76% of people across the world use banking and credit services. Today, that's a little over 5.3 BILLION people that have both formally and informally employed contractual procedures to store capital. As the world becomes increasingly digital, new mechanisms to store capital have spun up. 2009 saw the groundbreaking launch of Bitcoin, a decentralized peer-to-peer network with the goal of publicizing transaction information and forgoing a centralized authority in the transfer and transit of capital. Since then, new blockchains have arisen. Ethereum, the second most popular blockchain platform launched in 2015 and is currently adopted by more than 400 million users. The corollary to traditional banking and financial contracts on the Ethereum blockchain are smart-contracts: self-executing contracts with agreement terms directly written into lines of code. These smart-contracts, especially ERC-20 tokens, have revolutionized the way we think about transactions and asset management in the digital era.

However, with great power comes great responsibility. As these tokens grow in popularity and usage, concerns about their security and the potential for misuse have also escalated. This burgeoning digital ecosystem, while offering immense opportunities, also presents a fertile ground for fraudulent schemes and scams. The ease of creating and deploying ERC-20 tokens, coupled with the anonymity and lack of regulation inherent in the blockchain space, has led to an increase in deceptive practices.

Therefore, understanding the intricacies of these digital contracts is crucial. This research aims to dissect the underlying mechanisms of ERC-20 tokens, particularly focusing on how their unique properties might be exploited for nefarious purposes. Through a meticulous exploratory case study of various smart-tokens, this study endeavors to unveil the specific functionalities and loopholes within these contracts that make them prone to manipulation. By doing so, it seeks to contribute to the development of more secure and trustworthy digital financial systems, ensuring that the promise of blockchain technology is not overshadowed by its potential perils.

ERC-20 tokens—a standard for Ethereum-based smart contracts—have become a cornerstone of digital financial transactions. However, alongside their rapid growth and adoption, concerns about their misuse in fraudulent schemes and scams have escalated. This research seeks to investigate the underlying mechanisms of ERC-20 tokens, particularly focusing on how they might be exploited for deceptive purposes. Employing a qualitative exploratory case study method, this study scrutinizes a selection of thirty smart-tokens that register below a 70% similarity score when assessed using a winnowing-BERT based similarity detection algorithm. The primary goal is to unravel the specific functionalities within these digital contracts that render them susceptible to manipulative practices.

Literature Review

Decentralized finance (DeFi) has gained significant traction in recent years, offering users novel financial services built on blockchain technology. However, this emerging ecosystem presents new challenges, particularly in the detection of scams and fraudulent activities. Traditional finance has made significant breakthroughs in scam identification systems, and this literature review aims to explore how these breakthroughs can be integrated with contract identification systems for Layer-two Ethereum smart contracts to advance the detection of honeypot and general ERC-token scams.

Document similarity measurement is a technique employed in traditional finance's scam identification systems. By comparing the textual content of documents, such as contracts or transaction records, similarities and patterns indicative of scams can be identified. This approach can be applied to the detection of scams in Layer-two Ethereum smart contracts, where similarities between contracts can reveal potential fraudulent activity. Semantic embedding, a technique used to represent the meaning of text, is another breakthrough in scam identification systems. By capturing the semantic relationships between words or phrases, semantic embedding can provide a deeper understanding of document content and facilitate the identification of fraudulent patterns.

Contract identification systems for Ethereum smart contracts have made significant progress in detecting potential scams through the use of clustering, byte code analysis, and similarity matching techniques. These systems carefully analyze the bytecode of smart contracts deployed on the Ethereum blockchain, leveraging the structural and behavioral characteristics of the code to identify patterns and similarities that may indicate fraudulent activities. By examining the underlying structure of smart contracts, these systems can identify potential honeypot and general ERC-token scams.

Methods

An exploratory case study method is used to evaluate a sample of thirty smart-tokens that qualify below a 70% similarity score using a winnowing-BERT based similarity detection algorithm. Smart contracts, particularly ERC-20 tokens, are at the heart of the research question, which seeks to explore how these digital contracts implement mechanisms that could enable scams and fraudulent activity. Overall, the methods facilitate a dually-quantitative/qualitative approach towards understanding token mechanisms that enable scams.

Quantitative Methods

Utilizing a similarity checker tool offers an objective way to measure and compare token quantities against identified scam-types. The experiment can yield quantitative data showing the prevalence of contracts that may be associated with scams, such as "lazy-rug pulls." This approach allows for testing hypotheses regarding the frequency and characteristics of potentially fraudulent contracts in the sample. The quantitative method is determined through counting contracts that meet a benchmark for predefined scam characteristics. This is done through comparing against a 10k false honeypot checked data-set and winnowing similarity.

Qualitative Methods

In-depth code analysis provides qualitative insights into the contract types, revealing specific coding patterns or functions commonly seen in scams. The non-experimental design doesn't test a hypothesis but rather describes the features of smart contracts that could be indicative of fraudulent intent. Deep dives into code can reveal the intricacies and mechanisms within the contracts that may not be apparent through purely quantitative methods. The findings from the code analysis can contribute to a set of best practices or red flags for evaluating smart contracts in the future.

Building the Similarity Analytics Algorithm

To replicate the codebase for this research project, researchers should start by importing key dependencies and configuration files. The core of the codebase revolves around the masterChecker function, which processes tokens based on parameters such as token_address, pair_address, and current_iteration. This function conducts a series of evaluations, including decimal, verification, contract, and other relevant checks, updating the database with the results.

For blockchain interaction, the ethers library is used to establish a WebSocket connection with the Ethereum blockchain. This connection facilitates listening for new token pair creation events on the blockchain, using the Uniswap V2 Factory's address and event signature. Upon detection of new pairs, the code captures and stores relevant token data in JSON format within a specified directory, and updates the database accordingly.

Results

In the study's results section, histograms were used to visually represent the similarity scores of ERC-20 token codes. Each histogram corresponds to a different token, identified by its unique address, and illustrates the distribution of similarity scores across a sample set of smart contracts. The varying shapes and peaks of these histograms indicate how similar or different each token's code is when compared to others. For example, some tokens show a high frequency of certain similarity scores, suggesting common patterns or potential replication of code, which may raise concerns about originality or the presence of standardized coding practices. Other tokens demonstrate a broader spread of similarity scores, indicating a more diverse range of coding approaches. These visual representations provide insights into the prevalence of code similarities that could signify either widespread use of common coding patterns or possible vulnerabilities and risks associated with the ERC-20 tokens.

Discussion

The study's findings, presented through a series of histograms, reveal a range of code similarity scores among various ERC-20 tokens. These distributions provided insights into the prevalence of common coding patterns and potential copycat behavior, suggesting areas of both concern and interest for those maintaining blockchain security and integrity. A notable limitation of the research is the potential for selection bias in the sample of smart contracts analyzed. The ERC-20 tokens were chosen based on their similarity scores, which may not have been as random as required to ensure a comprehensive overview of the entire blockchain landscape. Additionally, the study's reliance on automated code similarity tools could overlook the nuanced context in which similar code structures are employed, thus potentially misidentifying benign standard practices as risks. The findings can be used to guide developers and security experts in scrutinizing ERC-20 contracts, promoting the adoption of more robust coding practices, and enhancing preventive measures against scams.

Conclusion

The research aimed to explore the extent of code similarity among ERC-20 tokens to identify potential security risks and vulnerabilities that could be exploited for fraudulent purposes. The study's results underscored a diverse landscape of code similarities, with some tokens displaying concerning patterns that warrant further scrutiny. Future research could focus on expanding the sample size to include a broader and more random selection of ERC-20 tokens, providing a more representative assessment of the ecosystem. Additionally, incorporating human-led code reviews could complement automated tools, offering deeper insights into the context and intention behind code similarities. Further studies could also explore the development of advanced tools for real-time monitoring of smart contracts, thereby providing more dynamic and responsive measures to secure the blockchain against the ever-evolving threat of scams.

Combined References

Part One: Frankenstein Media Analysis

Cambra-Badii, I., Guardiola, E., & Baños, J. E. (2021). Frankenstein; or, the modern Prometheus: A classic novel to stimulate the analysis of complex contemporary issues in biomedical sciences. *BMC Medical Ethics, 22*(1), 17.

Corrêa, N. K., et al. (2023). Worldwide AI ethics: A review of 200 guidelines and recommendations for AI governance. *Patterns, 4*(10), 100857.

European Parliament. (2024). *Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence*. Official Journal of the European Union.

Kant, I. (1993). *Grounding for the metaphysics of morals* (J. W. Ellington, Trans.; 3rd ed.). Hackett. (Original work published 1785)

Shelley, M. (2018). *Frankenstein: Annotated for scientists, engineers, and creators of all kinds*. MIT Press. (Original work published 1818)

Part Two: ERC-20 Smart Contract Research

Dagan, I., Karov, Y., & Roth, D. (1997). Mistake-driven learning in text categorization. *arXiv preprint cmp-lg/9706006*.

Liu, Z. (2023). Binary code similarity detection. *Swinburne University of Technology*. arXiv:2308.02992.

Reiff, N. (2019). What is ERC-20 and what does it mean for Ethereum? *Investopedia*.

TheGlobalEconomy.com. (n.d.). Percent people with bank accounts by country.

Zakeri-Nasrabadi, M. (2023). A systematic literature review on source code similarity measurement and clone detection. *Iran University of Science and Technology*. arXiv:2306.16171.

How can a recommender system identify and mitigate exploitable vulnerabilities in Ethereum’s ERC20 smart contracts to enhance blockchain security?

April 29th, 2024

Word Count: 4185

ABSTRACT

This research paper investigates the security vulnerabilities and potential for misuse of ERC-20 tokens, a standard for Ethereum-based smart contracts that have become integral to digital financial transactions. With over 5.3 billion people worldwide using banking and credit services and the rise of decentralized finance (DeFi), the advent of cryptocurrencies and smart contracts has revolutionized asset management and transactions. However, the growth of ERC-20 tokens has also raised concerns about their security and the increase in fraudulent schemes and scams. This study employs a dual quantitative and qualitative approach, including a qualitative exploratory case study of thirty smart-tokens and the development of the Similarity Analytics Algorithm, to examine the mechanisms of ERC-20 tokens and identify specific functionalities and loopholes that make them susceptible to exploitation. By analyzing these digital contracts' intricacies, the research aims to contribute to the development of more secure and trustworthy digital financial systems and enhance the integrity of blockchain technology.

INTRODUCTION

65.76% of people across the world use banking and credit services [1]. Today, that’s a little over 5.3 BILLION people that have both formally and informally employed contractual procedures to store capital [2]. As the world becomes increasingly digital, new mechanisms to store capital have spun up. 2009 saw the groundbreaking launch of Bitcoin, a decentralized peer-to-peer network with the goal of publicizing transaction information and forgoing a centralized authority in the transfer and transit of capital. Since then, new blockchains have arisen. Ethereum, the second most popular blockchain platform launched in 2015 and is currently adopted by more than 400 million users [3]. The corollary to traditional banking and financial contracts on the Ethereum blockchain are smart-contracts: self-executing contracts agreement terms directly written into lines of code. These smart-contracts, especially ERC-20 tokens, have revolutionized the way we think about transactions and asset management in the digital era. ERC-20 tokens, essentially digital assets built on Ethereum's blockchain, have become fundamental to a wide array of applications, from decentralized finance (DeFi) to non-fungible tokens (NFTs) [4]. They offer unparalleled flexibility and autonomy, allowing users to transact and interact in ways previously unimaginable.

However, with great power comes great responsibility. As these tokens grow in popularity and usage, concerns about their security and the potential for misuse have also escalated. This burgeoning digital ecosystem, while offering immense opportunities, also presents a fertile ground for fraudulent schemes and scams [5]. The ease of creating and deploying ERC-20 tokens, coupled with the anonymity and lack of regulation inherent in the blockchain space, has led to an increase in deceptive practices.

Therefore, understanding the intricacies of these digital contracts is crucial. This research aims to dissect the underlying mechanisms of ERC-20 tokens, particularly focusing on how their unique properties might be exploited for nefarious purposes. Through a meticulous exploratory case study of various smart-tokens, this study endeavors to unveil the specific functionalities and loopholes within these contracts that make them prone to manipulation. By doing so, it seeks to contribute to the development of more secure and trustworthy digital financial systems, ensuring that the promise of blockchain technology is not overshadowed by its potential perils.

ERC-20 tokens - a standard for Ethereum-based smart contracts - have become a cornerstone of digital financial transactions. However, alongside their rapid growth and adoption, concerns about their misuse in fraudulent schemes and scams have escalated. This research seeks to investigate the underlying mechanisms of ERC-20 tokens, particularly focusing on how they might be exploited for deceptive purposes. Employing an qualitative exploratory case study method, this study scrutinizes a selection of thirty smart-tokens that register below a 70% similarity score when assessed using a winnowing-BERT based similarity detection algorithm [6, 7]. The primary goal is to unravel the specific functionalities within these digital contracts that render them susceptible to manipulative practices.

The approach of this research is bifurcated into quantitative and qualitative methodologies. On the quantitative front, the study utilizes a similarity checker tool to objectively evaluate and compare the characteristics of tokens against known scam archetypes. This method facilitates the identification and measurement of contracts potentially linked to scams, such as the notorious "lazy-rug pulls" [7]. It allows for the testing of hypotheses related to the prevalence and attributes of potentially fraudulent contracts within the studied sample. Complementing this, the qualitative dimension of the study delves into an in-depth analysis of contract types. This involves a detailed examination of the coding patterns and functions that are commonly associated with scams. This level of analysis is critical to uncover the subtleties and complexities of smart contracts that may not be readily discernible through quantitative analysis alone.

Central to this investigation is the development of the Similarity Analytics Algorithm. This algorithm is pivotal to the study, incorporating advanced techniques in code similarity measurement and clone detection, as exemplified in the works of Morteza Zakeri-Nasrabadi, and binary code similarity detection, as proposed by Zian Liu [7, 8]. These methodologies have been adeptly tailored to suit the unique context of analyzing smart contracts. The research employs a sophisticated system architecture that dynamically interacts with the blockchain, specifically targeting the detection and analysis of new ERC-20 tokens. This is achieved through establishing a WebSocket connection with the Ethereum blockchain, which enables real-time monitoring and cataloging of new token events.

The process of this research is comprehensive and multi-layered. It begins with a crucial data preparation stage, where tokens displaying less than 70% similarity to established scam patterns are earmarked for further analysis [6, 7, 8]. This phase includes Website/White Paper Sentiment Analysis and Contract/Etherscan code analysis, crucial steps in evaluating the legitimacy of the projects [10, 11]. The culmination of this research is manifested in the creation of graphs and data analytics visualizations. These tools are instrumental in succinctly presenting the quantitative findings, offering insights into the distribution and characteristics of tokens that fall below the similarity threshold [8, 9]. This dataset is not only key in identifying potential scam tokens but also serves as a benchmark for future analyses in the rapidly evolving landscape of blockchain tokens.

Through this dual quantitative and qualitative approach, this research aspires to make a significant contribution to the understanding of token mechanisms that enable scams. It aims to provide valuable insights and tools for investors, developers, and regulators in the cryptocurrency domain, thereby aiding in the mitigation of fraud and enhancing the security and integrity of digital transactions in the blockchain ecosystem..

LITERATURE REVIEW

Decentralized finance (DeFi) has gained significant traction in recent years, offering users novel financial services built on blockchain technology. However, this emerging ecosystem presents new challenges, particularly in the detection of scams and fraudulent activities. Traditional finance has made significant breakthroughs in scam identification systems, and this literature review aims to explore how these breakthroughs can be integrated with contract identification systems for Layer-two Ethereum smart contracts to advance the detection of honeypot and general ERC-token scams. By enhancing security and trust in DeFi, these integrations can contribute to a more reliable and stable decentralized financial system. Traditional finance has made significant strides in the development of scam identification systems, incorporating various techniques to enhance their effectiveness. For example, machine learning algorithms have proven to be valuable in detecting fraudulent activities. As stated in [1], "We present an algorithm, a variation of Littlestone's Winnow, which performs significantly better than any other algorithm tested on this task using a similar feature set." This highlights the potential of machine learning to improve scam identification in the context of decentralized finance.

Document similarity measurement is another technique employed in traditional finance's scam identification systems. By comparing the textual content of documents, such as contracts or transaction records, similarities and patterns indicative of scams can be identified. [10] claims that "A visualization of specification coverage based on document similarity has been developed, which serves as a tool for the identification and resolution of gaps in service specification scenarios related to departments involved in service provision." This approach can be applied to the detection of scams in Layer-two Ethereum smart contracts, where similarities between contracts can reveal potential fraudulent activity. Semantic embedding, a technique used to represent the meaning of text, is another breakthrough in scam identification systems. By capturing the semantic relationships between words or phrases, semantic embedding can provide a deeper understanding of document content and facilitate the identification of fraudulent patterns. According to [8]; "SEA-PS proposes a novel semantic embedding approach, which outperforms existing methods in measuring patent similarity." Integrating semantic embedding into contract identification systems for Ethereum smart contracts can enhance the system's ability to detect scams by exploring semantic similarities between contract structures and known scam patterns. By integrating these breakthrough techniques from centralized finance's scam identification systems into contract identification systems for Layer-two Ethereum smart contracts, the detection of honeypot and general ERC-token scams can be significantly advanced. These integration efforts can improve the security and trust in decentralized finance by reducing the risk of fraudulent activities, protecting users from financial loss, and enhancing the overall resilience of the DeFi ecosystem.

Contract identification systems for Ethereum smart contracts have made significant progress in detecting potential scams through the use of clustering, byte code analysis, and similarity matching techniques [7][18]. These systems carefully analyze the bytecode of smart contracts deployed on the Ethereum blockchain, leveraging the structural and behavioral characteristics of the code to identify patterns and similarities that may indicate fraudulent activities. For instance, as mentioned in [7], "Our evaluation of the quality of clustering relies on a ground truth of token and wallet contracts identified in earlier work. Our analysis is based on the bytecodes deployed on the main chain of Ethereum up to block 10.5 million, mined on July 21, 2020." By examining the underlying structure of smart contracts, these systems can identify potential honeypot and general ERC-token scams. Integrating these contract identification systems with the scam identification systems from traditional finance can create a comprehensive detection framework for Layer-two Ethereum smart contracts. This combination leverages the strengths of both approaches, providing a more robust and accurate detection mechanism. By integrating the analysis of bytecode patterns and structural similarities with machine learning algorithms, document similarity measurement, and semantic embedding techniques, the detection framework can effectively identify fraudulent activities in smart contracts. The clustering technique used in contract identification systems enables the grouping of similar contracts, allowing for the identification of clusters associated with potential scams. By analyzing the behaviors and characteristics of contracts within these clusters, suspicious activities and patterns can be flagged for further investigation. This approach complements the scam identification systems in traditional finance, which also utilize clustering methodologies to identify patterns of fraudulent activities [3]. Combining the expertise and knowledge from both domains can enhance the accuracy and efficiency of scam detection in Layer-two Ethereum smart contracts. Furthermore, byte code analysis plays a crucial role in contract identification systems, as it enables the examination of low-level instructions and operations within smart contracts. This analysis allows for the identification of specific code patterns that are indicative of potential scams, such as the presence of malicious functions or suspicious transaction flows. By integrating this byte code analysis with the techniques used in traditional finance's scam identification systems, the detection framework can effectively capture the nuances and intricacies of fraudulent activities specific to decentralized finance. Similarity matching techniques provide another layer of analysis in contract identification systems. By comparing the bytecode of different smart contracts, similarities in structural patterns or functions can be identified. This approach can be further enhanced by leveraging document similarity measurement techniques from traditional finance's scam identification systems. These techniques enable the identification of similarities in contract content, such as overlapping sections or code snippets, even if the bytecode has undergone slight modifications. This ability to detect similarities in modified contracts can be crucial in spotting variations of known scam patterns.

In summary, integrating contract identification systems for Layer-two Ethereum smart contracts with the breakthroughs in traditional finance's scam identification systems brings together the strengths of both approaches. This integration creates a comprehensive detection framework that combines bytecode analysis, clustering, similarity matching, machine learning algorithms, document similarity measurement, and semantic embedding. By leveraging the expertise and techniques from both domains, this framework improves the security and trust in decentralized finance by effectively identifying and mitigating honeypot and general ERC-token scams. The integration of breakthroughs from centralized finance's scam identification systems with contract identification systems for Layer-two Ethereum smart contracts can have significant implications for enhancing security and trust in DeFi. Firstly, by leveraging machine learning algorithms and similarity measurement techniques, the detection of honeypot and ERC-token scams can be automated, allowing for real-time monitoring and prevention of fraudulent activities [1][2][16][19]. This automation ensures that scams can be quickly identified and mitigated, reducing the potential financial losses for users. Secondly, the integration of semantic embedding and document similarity measurement techniques can enhance the accuracy and efficiency of scam detection in smart contracts [8][12][13][15]. By considering the semantic similarities between contracts and known scam patterns, the system can identify potential scams even if they have slightly different bytecode structures. This approach improves the resilience of the detection system, as scammers may constantly modify their scam contracts to evade detection. Furthermore, the integration of contract and scam identification systems can provide valuable insights into the types of scams prevalent in the DeFi ecosystem [3][5][7][11][17][22]. By analyzing the patterns and characteristics of detected scams, regulators and developers can better understand the vulnerabilities in the system and implement measures to prevent future scams. This knowledge can contribute to the overall security and stability of DeFi.

The breakthroughs in centralized, traditional finance's scam identification systems can be integrated with contract identification systems for Layer-two Ethereum smart contracts to advance the detection of honeypot and general ERC-token scams. This integration has the potential to significantly enhance security and trust in decentralized finance. By leveraging machine learning algorithms, similarity measurement techniques, semantic embedding, and document similarity metrics, scams can be detected in real-time, providing users with a more secure and reliable DeFi ecosystem. Furthermore, the insights gained from analyzing detected scams can inform the development of preventive measures and contribute to the overall resilience of decentralized finance. Despite the progress made in integrating breakthroughs from centralized finance's scam identification systems with contract identification systems for Layer-two Ethereum smart contracts, there are still certain gaps that need to be addressed. One of the challenges is the rapidly evolving nature of scams in the decentralized finance space. Scammers are constantly finding new ways to exploit vulnerabilities and deceive users. Therefore, it is crucial to continuously update and adapt the detection framework to stay ahead of emerging scams. Another gap is the need for comprehensive and reliable datasets for training and testing the integrated scam detection systems. The lack of publicly accessible data-sets specific to Layer-two Ethereum smart contracts hinders the development and evaluation of effective detection models. To bridge this gap, collaborations between researchers, regulators, and industry stakeholders are essential to gather and share relevant data while ensuring privacy and security. Furthermore, addressing the scalability and efficiency of scam detection in Layer-two Ethereum smart contracts is crucial for wide-scale adoption. As the DeFi ecosystem continues to grow, the number and complexity of smart contracts increase exponentially. The integration of centralized finance's scam identification systems should consider the scalability requirements and leverage optimization techniques to process and analyze a large number of contracts in real-time. Lastly, establishing effective mechanisms for collaboration and information sharing between centralized finance institutions and decentralized finance communities is vital. This collaboration can help create a unified approach to detecting and preventing scams, leveraging the expertise and resources from both domains. Sharing knowledge, best practices, and lessons learned can foster a stronger and more secure decentralized financial system. Addressing these gaps will require ongoing research, collaboration, and advancements in technology. By actively working towards closing these gaps, the integration of breakthroughs from centralized, traditional finance's scam identification systems with contract identification systems for Layer-two Ethereum smart contracts can lead to significant advancements in the detection of scams and enhance the security and trust in decentralized finance.

METHODS SECTION

An exploratory case study method is used to evaluate a sample of thirty smart-tokens that qualify below a 70% similarity score using a winnowing-BERT based similarity detection algorithm. Smart contracts, particularly ERC-20 tokens, are at the heart of the research question, which seeks to explore how these digital contracts implement mechanisms that could enable scams and fraudulent activity. Overall, the methods facilitate a dually-quantitative/qualitative approach towards understanding token mechanisms that enable scams.

QUANTITATIVE METHODS

Utilizing a similarity checker tool offers an objective way to measure and compare token quantities against identified scam-types. The experiment can yield quantitative data showing the prevalence of contracts that may be associated with scams, such as "lazy-rug pulls." This approach allows for testing hypotheses regarding the frequency and characteristics of potentially fraudulent contracts in the sample. Depending on the experiment's design, it could help infer causal relationships between contract characteristics and scam-like behaviors. The quantitative method is determined through counting contracts that meet a benchmark for predefined scam characteristics. This will be done through comparing against a 10k false honeypot checked data-set and winnowing similarity. Count analysis (or seeing specific numbers of failed tokens) allows for the direct quantification of contracts that align with specific scam categories. Establishing a systematic approach to categorize and count contracts based on predefined scam characteristics offers a method that can be replicated by other researchers for validation or further study.

QUALITATIVE METHODS

In-depth code analysis provides qualitative insights into the contract types, revealing specific coding patterns or functions commonly seen in scams. The non-experimental design doesn't test a hypothesis but rather describes the features of smart contracts that could be indicative of fraudulent intent. Deep dives into code can reveal the intricacies and mechanisms within the contracts that may not be apparent through purely quantitative methods. The findings from the code analysis can contribute to a set of best practices or red flags for evaluating smart contracts in the future.

Building the Similarity Analytics Algorithm

To replicate the codebase for this research project, researchers should start by importing key dependencies and configuration files. This includes a configuration file (`config.js`), various modules from a Database API (`DatabaseAPI.js`), such as `ErrorDB` and `TokenDB`, and the file system module (`fs`). Additionally, specific functions for token checks—like decimal, verification, contract, whale, liquidity, counter, and honey pot checks—need to be imported from their respective files within the `TokenChecks` directory. The core of the codebase revolves around the `masterChecker` function, which processes tokens based on parameters such as `token_address`, `pair_address`, and `current_iteration`. This function conducts a series of evaluations, including decimal, verification, contract, and other relevant checks, updating the database with the results.

Verification and relevant checks parallel the requirements outlined in past papers; namely, that of Morteza Zakeri-Nasrabadi & Zian Liu. Morteza Zakeri-Nasrabadi's source code similarity measurement and clone detection provides a comprehensive overview of various techniques used in code-similarity detection, including concepts like Levinshtein distance similarity. These distance similarity metrics are trivial to implement, but leveraged in specific token-checks allowing us to better understand metrics from this standardized perspective [8]. Zian Liu's work on binary code similarity detection, as detailed in their 2023 paper, emphasizes the importance of detecting code similarities at the binary level, particularly in mutated binary codes produced by different compiling options [7]. This research proposes a novel approach involving symbolic execution of binary code, extraction of symbolic values, and comparison of symbolic graph similarity. Their methodology resonates with the project's approach of analyzing smart contract codes and is deeply integrated into the token-check algorithm. The techniques used in Liu's research, especially the focus on subfunctions and instruction components, are adapted for analyzing the assembly-level code of smart contracts, or the ABI (abstract binary interface). This enhances our ability to detect scams by uncovering deeper similarities in contract behavior, which might not be apparent at the source code level.

For blockchain interaction, the ethers library is used to establish a WebSocket connection with the Ethereum blockchain.

This connection facilitates listening for new token pair creation events on the blockchain, using the Uniswap V2 Factory's address and event signature. Upon detection of new pairs, the code captures and stores relevant token data in JSON format within a specified directory, and updates the database accordingly. File processing and iteration management are handled through functions designed to increase the iteration count of tokens and remove them from tracking as needed. A cyclical process is implemented to regularly review and update files in the designated directory, adjusting iteration counts based on the time elapsed since detection.

Comprehensive error handling is crucial. The `masterChecker` function includes mechanisms to log errors and record them in the ErrorDB, providing details such as the token address, error location, and a descriptive error message. This structured approach allows researchers to efficiently monitor, analyze, and store data on new ERC-20 tokens, facilitating the study of their characteristics and potential scam-like behaviors. The system's overall architecture is designed to dynamically interact with blockchain data, focusing primarily on the detection and analysis of new ERC-20 tokens. By leveraging the WebSocket connection, the system effectively listens for new token events, capturing and cataloging them in real-time. This proactive approach ensures that the dataset remains current and relevant to ongoing blockchain activities.

PROCEDURE

The data preparation stage is vital for identifying tokens for further examination. Tokens with less than 70% similarity to known scam patterns, as determined by the similarity detection algorithm, are flagged and, if selected from the random sample, subjected to further analysis. This includes conducting Website/White Paper Sentiment Analysis, where the tone and language used in the project's documentation are scrutinized for indications of legitimacy or deception. Additionally, Contract/Etherscan code analysis is performed, involving a thorough examination of the token's smart contract code and activity on Etherscan for any irregularities or patterns typical of scams. The final step is the creation of graphs and data analytics visualizations, which succinctly encapsulate the study's quantitative findings. These visual tools are critical in illustrating the distribution and characteristics of tokens falling below the 70% similarity threshold, forming a comprehensive dataset that not only highlights potential scam tokens but also serves as a benchmark for ongoing blockchain token analysis.

RESULTS

In my study's results section, histograms were used to visually represent the similarity scores of ERC-20 token codes. Each histogram corresponds to a different token, identified by its unique address, and illustrates the distribution of similarity scores across a sample set of smart contracts. The varying shapes and peaks of these histograms indicate how similar or different each token's code is when compared to others. For example, some tokens show a high frequency of certain similarity scores, suggesting common patterns or potential replication of code, which may raise concerns about originality or the presence of standardized coding practices. Other tokens demonstrate a broader spread of similarity scores, indicating a more diverse range of coding approaches. These visual representations provide insights into the prevalence of code similarities that could signify either widespread use of common coding patterns or possible vulnerabilities and risks associated with the ERC-20 tokens. Further analysis is needed to interpret the implications of these distributions for the security and integrity of the tokens involved.

DISCUSSION

The study's findings, presented through a series of histograms, reveal a range of code similarity scores among various ERC-20 tokens. These distributions provided insights into the prevalence of common coding patterns and potential copycat behavior, suggesting areas of both concern and interest for those maintaining blockchain security and integrity. A notable limitation of the research is the potential for selection bias in the sample of smart contracts analyzed. The ERC-20 tokens were chosen based on their similarity scores, which may not have been as random as required to ensure a comprehensive overview of the entire blockchain landscape. Additionally, the study's reliance on automated code similarity tools could overlook the nuanced context in which similar code structures are employed, thus potentially misidentifying benign standard practices as risks. The interpretative nature of the data requires a careful approach to avoid overestimating the prevalence of fraudulent behavior. The findings can be used to guide developers and security experts in scrutinizing ERC-20 contracts, promoting the adoption of more robust coding practices, and enhancing preventive measures against scams.

CONCLUSION

The research aimed to explore the extent of code similarity among ERC-20 tokens to identify potential security risks and vulnerabilities that could be exploited for fraudulent purposes. The study's results underscored a diverse landscape of code similarities, with some tokens displaying concerning patterns that warrant further scrutiny. Future research could focus on expanding the sample size to include a broader and more random selection of ERC-20 tokens, providing a more representative assessment of the ecosystem. Additionally, incorporating human-led code reviews could complement automated tools, offering deeper insights into the context and intention behind code similarities. Further studies could also explore the development of advanced tools for real-time monitoring of smart contracts, thereby providing more dynamic and responsive measures to secure the blockchain against the ever-evolving threat of scams.

References

[1] “Percent people with bank accounts by country, around the world,” *TheGlobalEconomy.com*. https://www.theglobaleconomy.com/rankings/percent_people_bank_accounts/

[2] Worldometer, “World Population Clock,” *Worldometers*, Jul. 16, 2023. https://www.worldometers.info/world-population/

[3] “The History of Ethereum: Its Origin and Upgrades | Worldcoin,” *worldcoin.org*. https://worldcoin.org/articles/history-of-ethereum

[4] N. Reiff, “What is ERC-20 and What Does it Mean for Ethereum?,” *Investopedia*, 2019. https://www.investopedia.com/news/what-erc20-and-what-does-it-mean-ethereum/

[5] Harry, “Bad Actors Abusing ERC20 Approval to Steal Your Tokens!,” *MyCrypto*, Feb. 25, 2021. https://medium.com/mycrypto/bad-actors-abusing-erc20-approval-to-steal-your-tokens-c0407b7f7c7c

[6] I. Dagan, Y. Karov, and D. Roth, "Mistake-Driven Learning in Text Categorization," 1997. [Online]. Available: https://arxiv.org/pdf/cmp-lg/9706006.pdf. [Accessed: Jan. 19, 2023].

[7] Z. Liu, "Binary Code Similarity Detection," in Proc. Department of Computer Science and Software Engineering, Swinburne University of Technology, 2023. [Online]. Available: https://arxiv.org/pdf/2308.02992.pdf

[8] M. Zakeri-Nasrabadi, "A systematic literature review on source code similarity measurement and clone detection: Techniques, applications, and challenges," in Proc. School of Computer Engineering, Iran University of Science and Technology, 2023, pp. 1-136. [Online]. Available: https://arxiv.org/pdf/2306.16171.pdf

[9] C. Huang, G. Zhu, G. Ge, T. Li, Z. Lab, and J. Wang, "FastBCSD: Fast and Efficient Neural Network for Binary Code Similarity Detection." [Online]. Available: https://arxiv.org/pdf/2306.14168.pdf. [Accessed: Jan. 19, 2023].

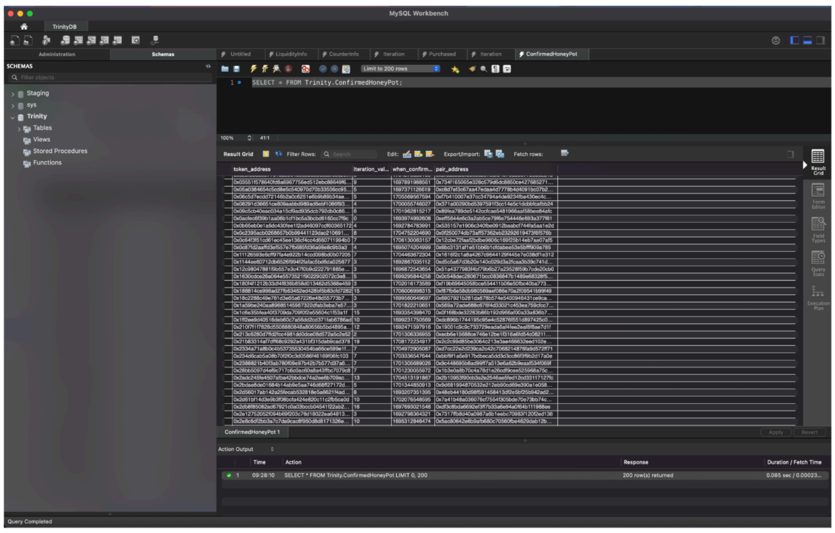
[10] H. Koo, S. Park, D. Choi, and T. Kim, "Semantic-aware Binary Code Representation with BERT." [Online]. Available: https://arxiv.org/pdf/2106.05478.pdf. [Accessed: Jan. 19, 2023].

[11] C. Tan and S. Yang, "Fine-grained Similarity Matching with a Similarity Filtration Pyramid for Code Search," in 2023 International Joint Conference on Neural Networks (IJCNN), Gold Coast, Australia, 2023, pp. 1-8, doi: 10.1109/IJCNN54540.2023.10191573. [Accessed: Jan. 19, 2023].

Appendix A

On this page you should have a link to, copy of, etc. the tool you used in order to conduct your study.

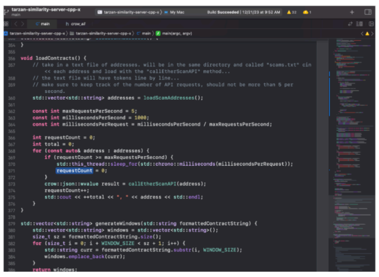
Data findings / my tools



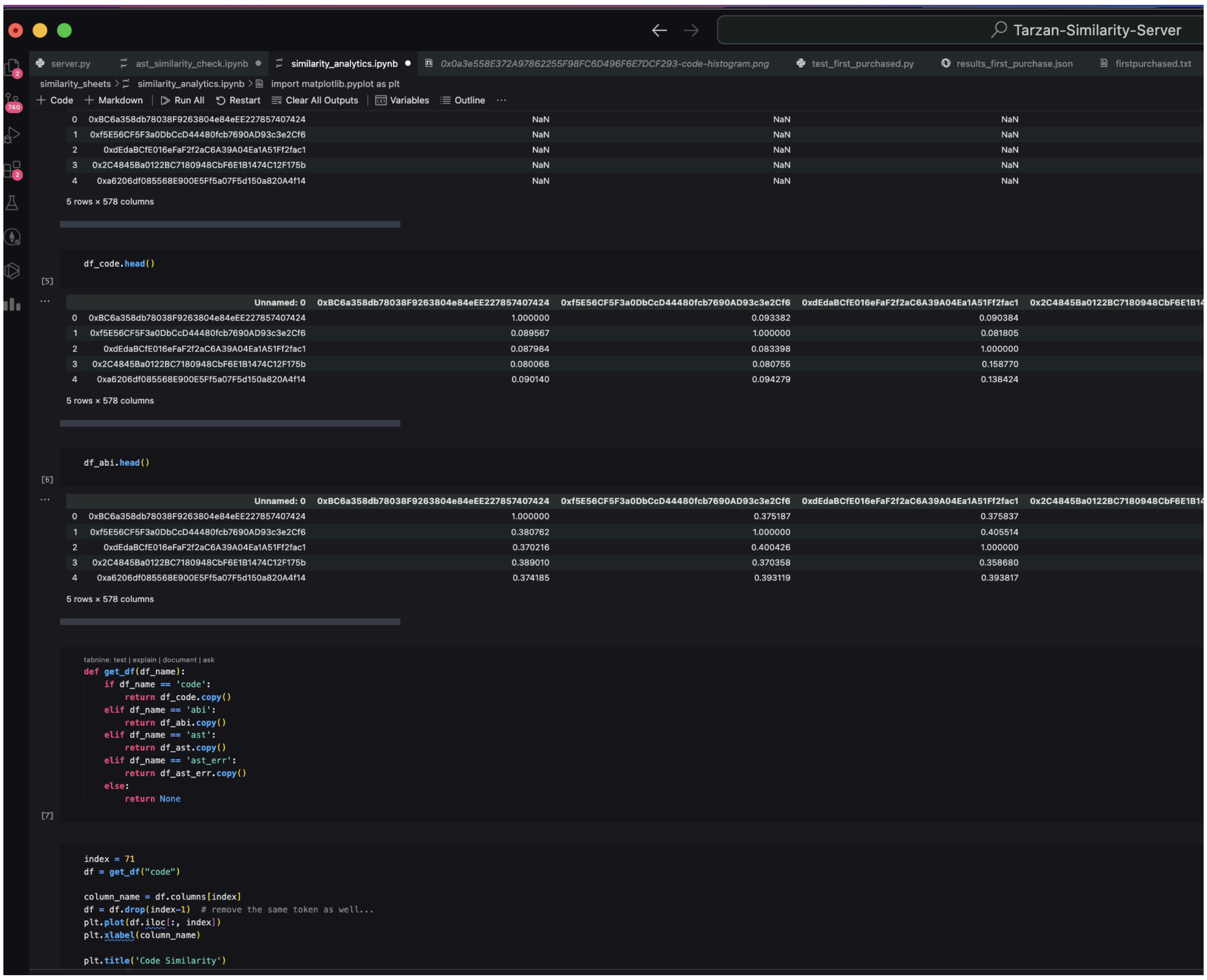
```
SELECT * FROM Trinity.ConfirmedHoneyPot;
```

token_address	iteration_val...	when_confirm...	pair_address
0x03551f578640fd8a6967756ed512ebc88649f6...	9	1697891988561	0x734f165065e328c579d6dc8850ce427685271...
0x05a0384654c5cd8e5c540970d70b33506cc95...	5	1697371126619	0xc8d7ef3c67aa47edaa4d7778b4d4091bc07b2...
0x06c5d7ecdd72146b2a0c6251e6b9b89b34ae...	5	1705569567594	0xf7b410007e37cc34794a4de9234fba430ec4c...
0x08291d36651ce809aabbd989ad6ebf1086f93...	4	1700055746027	0x371a00290bd5397591f3cc14e5c1dcbbfcafbb24
0x09c5cb40eac034a15cf9ad935dcb792db0c86...	6	1701962815217	0x89fea789de5142ccfcae5481966aaf58bee84afc
0x0acfec6f39b1aa06b1cf1bc5a3bcbd6160cc7f9c	0	1693974992608	0xeff5644e6c3a2ab5ce79f6e754446e693a377f61
0x0b65eb0e1a6dc430fee1f2ad46097cdf60365172	4	1692784783991	0x535157e1906c340fbe0912baabcf744fa5aa1e2d
0x0c2395acb0268657b0b99441123dac210691...	8	1704752204690	0x0f250074db73aff57362eb23292619473f6f576b
0x0c64f3f51cd61ec45ee136cf4cc4d660711994b0	7	1706130083157	0x12cbe72faaf2bdbe9606c169f25b14eb7ae07af5
0x0c87fd2aaffd3ef557e7fb685fd36a98e8c9b3a3	4	1695074204999	0x6bc3131af1e51bb6b1cfdabee53e5bfff909a785
0x11126593e6cff97fa4e922b14ccd098bd0b07205	7	1704463672304	0x1616f2c1a8a4267c9644129f445e7e038df1e312
0x1144ee60712db6526f994f2fafac5bd6da025677	3	1692867035112	0xd5c5a67d3b20e140c029d3a2fcaa3b39c741d...
0x12c98047881f9b557e3c47f0b9d222791885e...	3	1696872543654	0x51a4377983f4bf79b6b27a23528f59b7cde20cb0
0x1630cdce26a064e5573521f9022902072c3e8...	5	1699295844258	0x0c548dec280671bcc0836847b1489e68328f5...
0x180f4f1212b33df4f836b858d013462d5368e459	3	1702016173589	0xf19b69645058bce554411b06e50fbc40ba773...
0x188814ce996ad27fb63462ed428bf5b83cfd7282	15	1706006998315	0xf87fb6e58db980569aef086e70a2f09541b99f49
0x18c2288c49e761d3e65a67226e48d55773b7...	3	1699560649697	0x6907921b281da678b574e5400946431ce9ca...
0x1a59be240aa89685145967322dfab3eba7e57...	3	1701822210651	0x569a72ade688c676f4d33021c453ea759cfcc7...
0x1c6e35bfea40f3709da709f0f2e55604c1f53a1f	15	1693354398470	0x0f168bde32283b86b192d966af00a33a836b7...
0x1ff2ee9d40516deb60c7a56dd2cd371fab6786ad	10	1699231750569	0xdc896b1744195c95e4c52876f551d897425c0...
0x210f7f1f7828d5508880848a80656b5bd4895a...	12	1692471597916	0x19001c9c9c733729eada6af4fee2eaf8f8ae7d1f
0x213c6280d7ffd2fcc4981dd0dce08d572a5c2e52	2	1701306336955	0xecb6e15688ce746e12be1f316a6fd54c08211...
0x21b83314af7dff68c9292a431bf315dab9cad378	19	1708172234917	0x2c2c99d85be3064c213e3ae466632eed102e...
0x2334a71a8b0c4b53735530454ba66ce599e1f...	7	1704972905087	0xd7cc22e2d239ca2c42c7068214876fa9d572ff71
0x234d6cab5a08b70f2f0c3d0586f46169f06fc103	7	1703336547644	0xbbf9f1a6e917bdbeca5dd3d3cc86f3f9b2d17a0e
0x2388821b40f3ab780f09e97b42b7b577d37a6...	7	1701305689026	0x9c448695b8ad99ff7a513e6a62b9eaaf534f069f
0x28bb5097d4ef9c717c6c0ac60a8a43ffbc7079c8	7	1701230055972	0x1b3e0a8b70c4a78d1e26cdf9cee525968a75c...
0x2adc245fe4507afba42bbdce74a2ee6b709ac...	13	1704513191867	0x2b10953f90cb3a2e2546ae5fed12cd33117127fc
0x2bdae8de01684b14ab9e5aa746d68ff27172d...	5	1701344850913	0x9d681994870532e212eb90cd69e390a1e058...
0x2d56017ab142a25fecab532818e5a6621f4ad...	8	1693207351395	0x48eb44180d98f591458413df2e5bf25b942ad2...
0x2d61bf14d3e9b3f08bcfa424e820c11c2fb5ca0d	10	1702076548595	0x7a41b48a036076cf7554f305bde70e73bb74c...
0x2db8f85082ad67921c0a03bccb04541f22ab2...	16	1697693021548	0xdf3c8bda6692ef3ff7b33a6e94a0f64b111988ee
0x2e12752052f094b69f203c76d18022ea64813...	3	1692798364321	0x7317ffb8d40a0987a6b1eebc70983f120f2ed136
0x2e8c6df2bb3a7c7da9cac8f950d8d8171326e...	10	1695312846474	0x5ac80642e6b9afb680c70560fbe4629dab12b...

	Time	Action	Response	Duration / Fetch Time
1	09:28:10	SELECT * FROM Trinity.ConfirmedHoneyPot LIMIT 0, 200	200 row(s) returned	0.085 sec / 0.00023...


tarzan-similarity-server-cpp-x
main
tarzan-similarity-server-cpp-x
My Mac
Build Succeeded | 12/21/23 at 9:52 AM
32
1
main
crow_all
tarzan-similarity-server-cpp-x
tarzan-similarity-server-cpp-x
main
main(argc, argv)

```cpp
}

void loadContracts() {
    // take in a text file of addresses. will be in the same directory and called "scams.txt" cin
        << each address and load with the "callEtherScanAPI" method...
    // the text file will have tokens line by line...
    // make sure to keep track of the number of API requests, should not be more than 5 per
        second.
    std::vector<std::string> addresses = loadScamAddresses();

    const int maxRequestsPerSecond = 5;
    const int millisecondsPerSecond = 1000;
    const int millisecondsPerRequest = millisecondsPerSecond / maxRequestsPerSecond;

    int requestCount = 0;
    int total = 0;
    for (const auto& address : addresses) {
        if (requestCount >= maxRequestsPerSecond) {
            std::this_thread::sleep_for(std::chrono::milliseconds(millisecondsPerRequest));
            requestCount = 0;
        }
        crow::json::wvalue result = callEtherScanAPI(address);
        requestCount++;
        std::cout << ++total << ", " << address << std::endl;
    }
}

std::vector<std::string> generateWindows(std::string formattedContractString) {
    std::vector<std::string> windows = std::vector<std::string>();
    size_t sz = formattedContractString.size();
    for (size_t i = 0; i + WINDOW_SIZE < sz + 1; i++) {
        std::string curr = formattedContractString.substr(i, WINDOW_SIZE);
        windows.emplace_back(curr);
    }
    return windows;
```


Tarzan-Similarity-Server
server.py
ast_similarity_check.ipynb
similarity_analytics.ipynb
0x0a3e558E372A97862255F98FC6D496F6E7DCF293-code-histogram.png
test_first_purchased.py
results_first_purchase.json
firstpurchased.txt
similarity_sheets > similarity_analytics.ipynb > import matplotlib.pyplot as plt
+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ···

0	0xBC6a358db78038F9263804e84eEE227857407424	NaN	NaN	NaN
1	0xf5E56CF5F3a0DbCcD44480fcb7690AD93c3e2Cf6	NaN	NaN	NaN
2	0xdEdaBCfE016eFaF2f2aC6A39A04Ea1A51Ff2fac1	NaN	NaN	NaN
3	0x2C4845Ba0122BC7180948CbF6E1B1474C12F175b	NaN	NaN	NaN
4	0xa6206df085568E900E5Ff5a07F5d150a820A4f14	NaN	NaN	NaN

5 rows × 578 columns

```
df_code.head()
```

[5]

	Unnamed: 0	0xBC6a358db78038F9263804e84eEE227857407424	0xf5E56CF5F3a0DbCcD44480fcb7690AD93c3e2Cf6	0xdEdaBCfE016eFaF2f2aC6A39A04Ea1A51Ff2fac1	0x2C4845Ba0122BC7180948CbF6E1B14
0	0xBC6a358db78038F9263804e84eEE227857407424	1.000000	0.093382	0.090384	
1	0xf5E56CF5F3a0DbCcD44480fcb7690AD93c3e2Cf6	0.089567	1.000000	0.081805	
2	0xdEdaBCfE016eFaF2f2aC6A39A04Ea1A51Ff2fac1	0.087984	0.083398	1.000000	
3	0x2C4845Ba0122BC7180948CbF6E1B1474C12F175b	0.080068	0.080755	0.158770	
4	0xa6206df085568E900E5Ff5a07F5d150a820A4f14	0.090140	0.094279	0.138424	

5 rows × 578 columns

```
df_abi.head()
```

[6]

	Unnamed: 0	0xBC6a358db78038F9263804e84eEE227857407424	0xf5E56CF5F3a0DbCcD44480fcb7690AD93c3e2Cf6	0xdEdaBCfE016eFaF2f2aC6A39A04Ea1A51Ff2fac1	0x2C4845Ba0122BC7180948CbF6E1B14
0	0xBC6a358db78038F9263804e84eEE227857407424	1.000000	0.375187	0.375837	
1	0xf5E56CF5F3a0DbCcD44480fcb7690AD93c3e2Cf6	0.380762	1.000000	0.405514	
2	0xdEdaBCfE016eFaF2f2aC6A39A04Ea1A51Ff2fac1	0.370216	0.400426	1.000000	
3	0x2C4845Ba0122BC7180948CbF6E1B1474C12F175b	0.389010	0.370358	0.358680	
4	0xa6206df085568E900E5Ff5a07F5d150a820A4f14	0.374185	0.393119	0.393817	

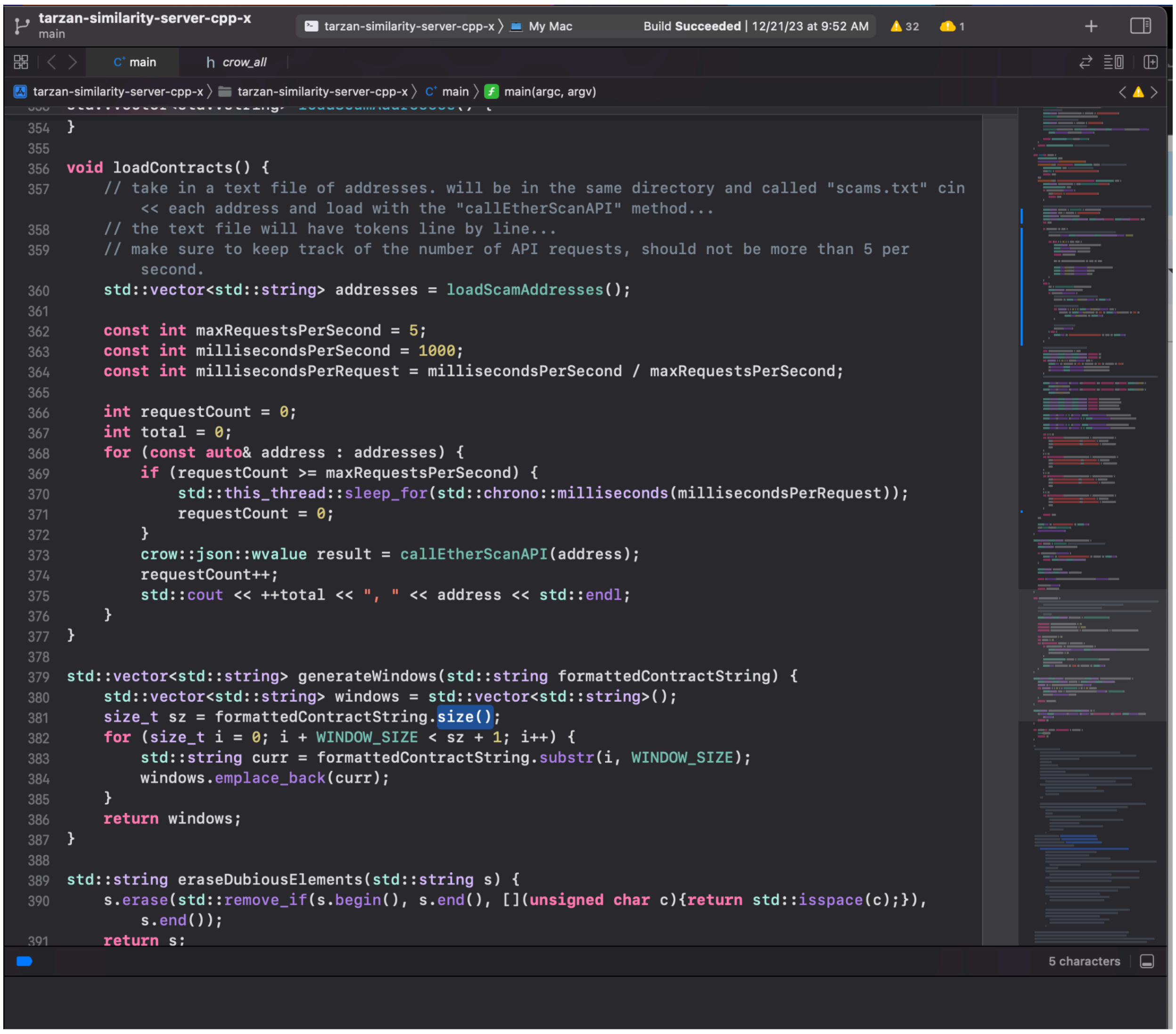
5 rows × 578 columns

tabnine: test | explain | document | ask

```
def get_df(df_name):
    if df_name == 'code':
        return df_code.copy()
    elif df_name == 'abi':
        return df_abi.copy()
    elif df_name == 'ast':
        return df_ast.copy()
    elif df_name == 'ast_err':
        return df_ast_err.copy()
    else:
        return None
```

[7]

```
index = 71
df = get_df("code")

column_name = df.columns[index]
df = df.drop(index-1)  # remove the same token as well...
plt.plot(df.iloc[:, index])
plt.xlabel(column_name)

plt.title('Code Similarity')
```

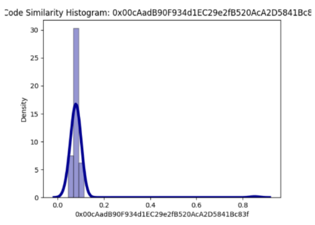

tarzan-similarity-server-cpp-x
main
tarzan-similarity-server-cpp-x 〉 My Mac
Build Succeeded | 12/21/23 at 9:52 AM
32
1
main
crow_all
tarzan-similarity-server-cpp-x 〉 tarzan-similarity-server-cpp-x 〉 main 〉 main(argc, argv)
5 characters

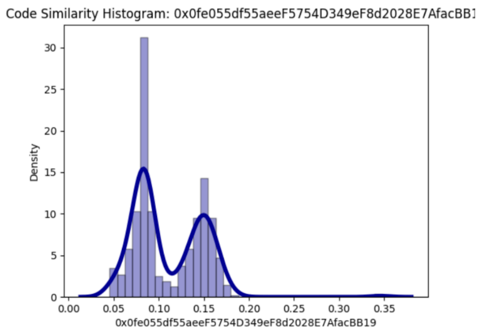
```cpp
}

void loadContracts() {
    // take in a text file of addresses. will be in the same directory and called "scams.txt" cin << each address and load with the "callEtherScanAPI" method...
    // the text file will have tokens line by line...
    // make sure to keep track of the number of API requests, should not be more than 5 per second.
    std::vector<std::string> addresses = loadScamAddresses();

    const int maxRequestsPerSecond = 5;
    const int millisecondsPerSecond = 1000;
    const int millisecondsPerRequest = millisecondsPerSecond / maxRequestsPerSecond;

    int requestCount = 0;
    int total = 0;
    for (const auto& address : addresses) {
        if (requestCount >= maxRequestsPerSecond) {
            std::this_thread::sleep_for(std::chrono::milliseconds(millisecondsPerRequest));
            requestCount = 0;
        }
        crow::json::wvalue result = callEtherScanAPI(address);
        requestCount++;
        std::cout << ++total << ", " << address << std::endl;
    }
}

std::vector<std::string> generateWindows(std::string formattedContractString) {
    std::vector<std::string> windows = std::vector<std::string>();
    size_t sz = formattedContractString.size();
    for (size_t i = 0; i + WINDOW_SIZE < sz + 1; i++) {
        std::string curr = formattedContractString.substr(i, WINDOW_SIZE);
        windows.emplace_back(curr);
    }
    return windows;
}

std::string eraseDubiousElements(std::string s) {
    s.erase(std::remove_if(s.begin(), s.end(), [](unsigned char c){return std::isspace(c);}), s.end());
    return s;
```

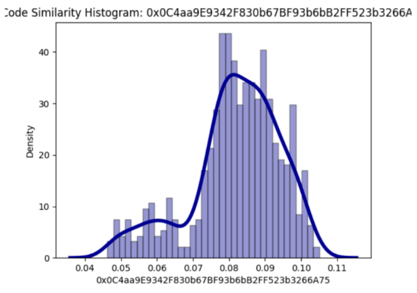
Appendix B

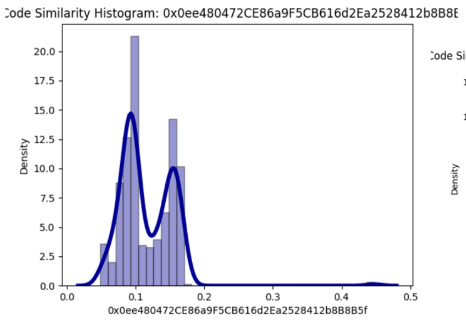
You can include charts and graphs of your data from the Results section of the report here.
Further information about my codified results below
https://docs.google.com/spreadsheets/d/1fiCFmtQBbaAUChto2YvonOXbQ5UsR7T0ljL9OzMlUrI/edit?usp=sharing

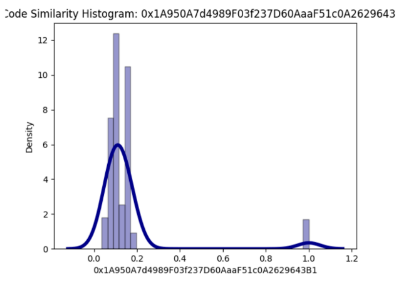
Data findings (individual tokens)

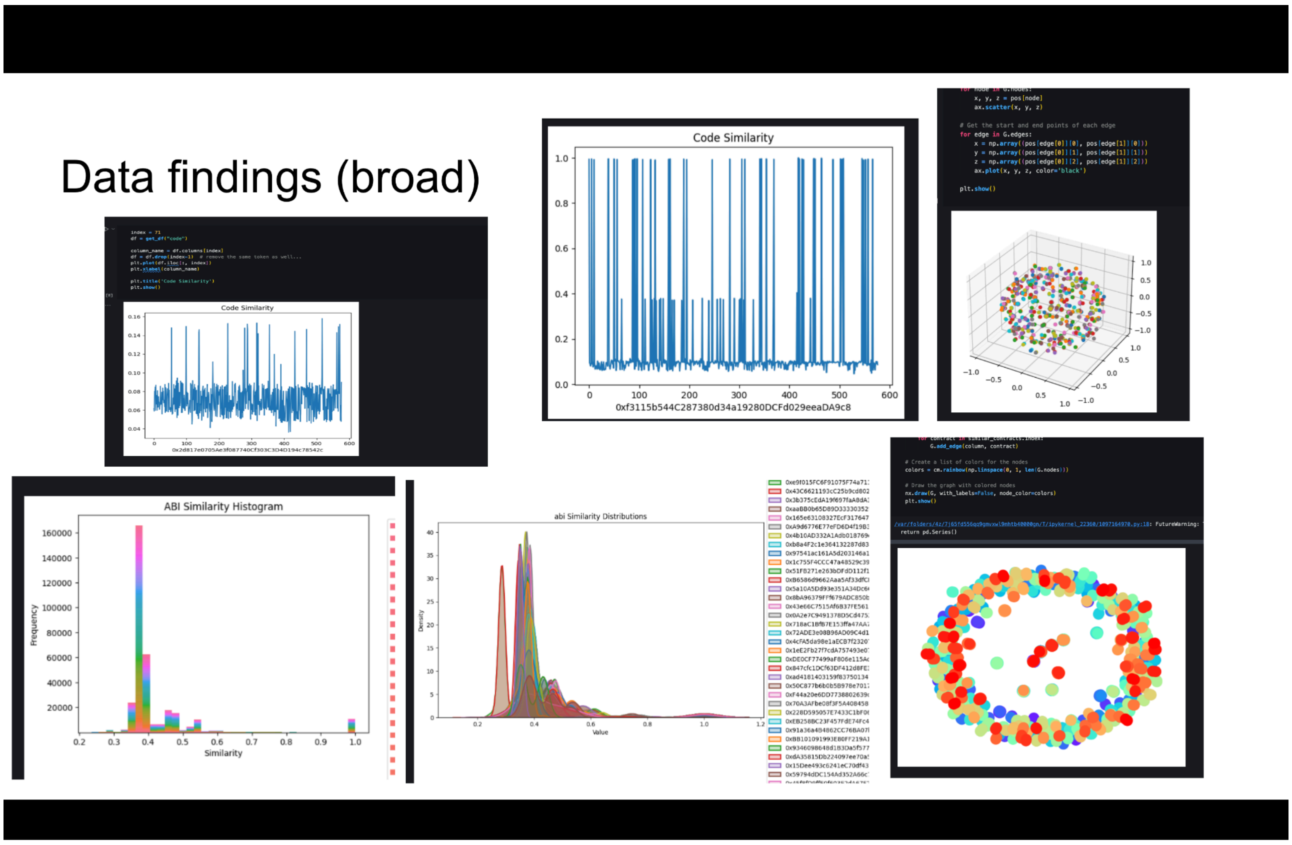












Data findings (broad)

```
index = 71
df = get_df("code")

column_name = df.columns[index]
df = df.drop(index-1)  # remove the same token as well...
plt.plot(df.iloc[:, index])
plt.xlabel(column_name)

plt.title('Code Similarity')
plt.show()
```

```
    x, y, z = pos[node]
    ax.scatter(x, y, z)

# Get the start and end points of each edge
for edge in G.edges:
    x = np.array((pos[edge[0]][0], pos[edge[1]][0]))
    y = np.array((pos[edge[0]][1], pos[edge[1]][1]))
    z = np.array((pos[edge[0]][2], pos[edge[1]][2]))
    ax.plot(x, y, z, color='black')

plt.show()
```

```
        G.add_edge(column, contract)

# Create a list of colors for the nodes
colors = cm.rainbow(np.linspace(0, 1, len(G.nodes)))

# Draw the graph with colored nodes
nx.draw(G, with_labels=False, node_color=colors)
plt.show()
```

```
/var/folders/4z/7j65fd556qq9gmvxwl9nhtb40000gn/T/ipykernel_22360/1097164970.py:18: FutureWarning:
  return pd.Series()
```

Code Similarity
0x2d817e0705Ae3f087740Cf303C3D4D194c78542c
Code Similarity
0xf3115b544C287380d34a19280DCFd029eeaDA9c8
ABI Similarity Histogram
Frequency
Similarity
abi Similarity Distributions
Density
Value

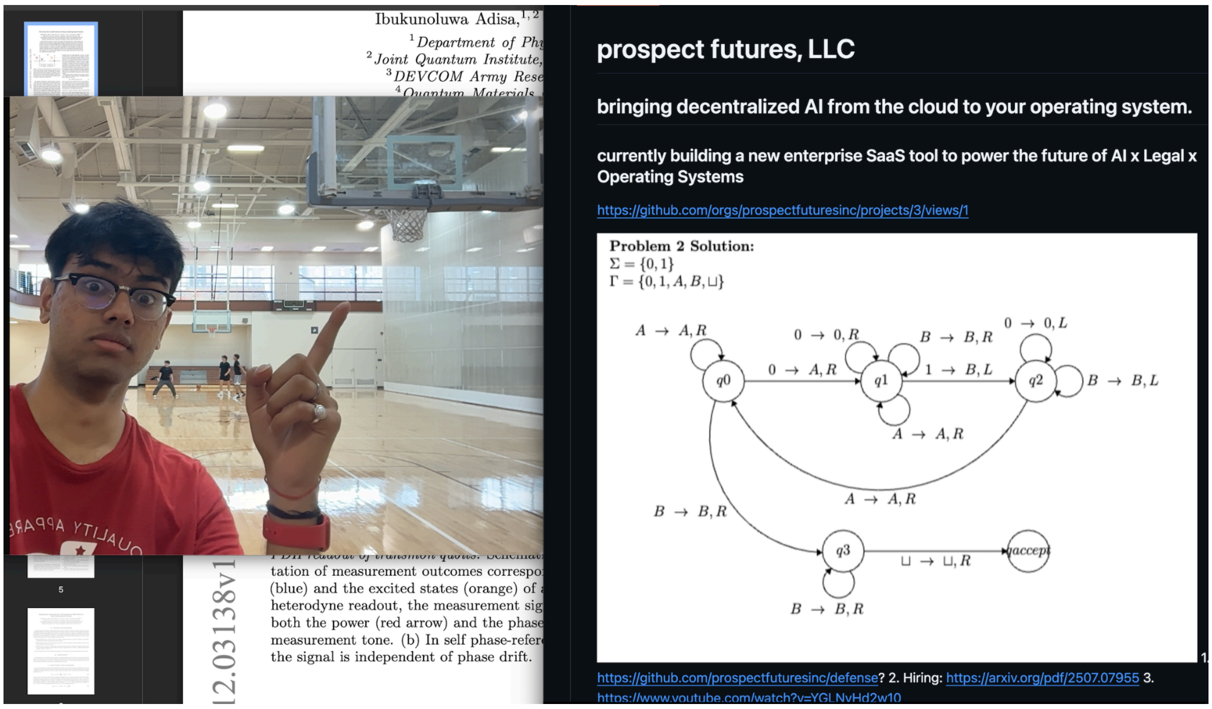
HonePot Table

token_address	iteration_value	when_confirmed	pair_address
0x017652d213e3cf1a1b2e22daad1ac30048190117	2	1698193482827	0x302da463a13f0966f6e186c5a84e7e14e0845b4e
0x02bf38889277e1c329643ec9b858c5acf69c2225	5	1704273380109	0xbf6a9bafbf8863c2072df54e76d590772eb62b1c
0x03551f578640fd8a6967756ed512ebc88649f6a6	9	1697891988561	0x734f165065e328c579d6dc8850ce427685271bf3
0x05a0384654c5cd8e5c540970d70b33506cc95f0b	5	1697371126619	0xc8d7ef3c67aa47edaa4d7778b4d4091bc07b23a3
0x06c5d7ecdd72146b2a0c6251e6b9b89b34ae05ce	5	1705569567594	0xf7b410007e37cc34794a4de9234fba430ec4cec2
0x08291d366	4	1700055746027	0x371a00290bd5397591f3cc14e5c1dcbbfcafbb24

51ce809aabbd989ad6ebf1086f931a4			
0x09c5cb40eac034a15cf9ad935dcb792db0c86a8b	6	1701962815217	0x89fea789de5142ccfcae5481966aaf58bee84afc
0x0acfec6f39b1aa06b1cf1bc5a3bcbd6160cc7f9c	0	1693974992608	0xeff5644e6c3a2ab5ce79f6e754446e693a377f61
0x0b65eb0e1a6dc430fee1f2ad46097cdf60365172	4	1692784783991	0x535157e1906c340fbe0912baabcf744fa5aa1e2d
0x0c2395acb0268657b0b99441123dac2106915ba2	8	1704752204690	0x0f250074db73aff57362eb23292619473f6f576b
0x0c64f3f51cd61ec45ee136cf4cc4d660711994b0	7	1706130083157	0x12cbe72faaf2bdbe9606c169f25b14eb7ae07af5
0x0c87fd2aaffd3ef557e7fb685fd36a98e8c9b3a3	4	1695074204999	0x6bc3131af1e51bb6b1cfdabee53e5bfff909a785
0x11126593e6cff97fa4e922b14ccd098bd0b07205	7	1704463672304	0x1616f2c1a8a4267c9644129f445e7e038df1e312
0x1144ee60712db6526f994f2fafac5bd6da025677	3	1692867035112	0xd5c5a67d3b20e140c029d3a2fcaa3b39c741de02
0x12c98047881f9b557e3c47f0b9d222791885e024	3	1696872543654	0x51a4377983f4bf79b6b27a23528f59b7cde20cb0
0x1630cdce26a064e5573521f9022902072c3e823b	5	1699295844258	0x0c548dec280671bcc0836847b1489e68328f5498
0x180f4f1212b33df4f836b858d013462d5368e459	3	1702016173589	0xf19b69645058bce554411b06e50fbc40ba773e97

0x188814ce996ad27fb63462ed428bf5b83cfd7282	15	1706006998315	0xf87fb6e58db980569aef086e70a2f09541b99f49
0x18c2288c49e761d3e65a67226e48d55773b7b6b2	3	1699560649697	0x6907921b281da678b574e5400946431ce9ca9455
0x1a59be240aa89685145967322dfab3eba7e574db	3	1701822210651	0x569a72ade688c676f4d33021c453ea759cfcc786
0x1c6e35bfea40f3709da709f0f2e55604c1f53a1f	15	1693354398470	0x0f168bde32283b86b192d966af00a33a836b725c
0x1ff2ee9d40516deb60c7a56dd2cd371fab6786ad	10	1699231750569	0xdc896b1744195c95e4c52876f551d897425c0cbe
0x210f7f1f7828d5508880848a80656b5bd4895ad8	12	1692471597916	0x19001c9c9c733729eada6af4fee2eaf8f8ae7d1f
0x213c6280d7ffd2fcc4981dd0dce08d572a5c2e52	2	1701306336955	0xecb6e15688ce746e12be1f316a6fd54c08211e28
0x2334a71a8b0c4b53735530454ba66ce599e1fd11	7	1704972905087	0xd7cc22e2d239ca2c42c7068214876fa9d572ff71
0x234d6cab5a08b70f2f0c3d0586f46169f06fc103	7	1703336547644	0xbbf9f1a6e917bdbeca5dd3d3cc86f3f9b2d17a0e
0x2388821b40f3ab780f09e97b42b7b577d37a6d5e	7	1701305689026	0x9c448695b8ad99ff7a513e6a62b9eaaf534f069f
0x28bb5097d4ef9c717c6c0ac60a8a43ffbc7079c8	7	1701230055972	0x1b3e0a8b70c4a78d1e26cdf9cee525968a75ce25
0x2adc245fe4507afba42bbdce74a2ee6b	13	1704513191867	0x2b10953f90cb3a2e2546ae5fed12cd33117127fc

709ac3a8			
0x2bdae8de01684b14ab9e5aa746d68ff27172d9b5	5	1701344850913	0x9d681994870532e212eb90cd69e390a1e0584874
0x2d56017ab142a25fecab532818e5a6621f4ad052	8	1693207351395	0x48eb44180d98f591458413df2e5bf25b942ad213
0x2d61bf14d3e9b3f08bcfa424e820c11c2fb5ca0d	10	1702076548595	0x7a41b48a036076cf7554f305bde70e73bb74cc73
0x2db8f85082ad67921c0a03bccb04541f22ab288c	16	1697693021548	0xdf3c8bda6692ef3ff7b33a6e94a0f64b111988ee
0x2e12752052f094b69f203c76d18022ea648137d1	3	1692798364321	0x7317ffb8d40a0987a6b1eebc70983f120f2ed136
0x2e8c6df2bb3a7c7da9cac8f950d8d8171326e701	10	1695312846474	0x5ac80642e6b9afb680c70560fbe4629dab12bc63
0x2fcbd5a6eb694d573d280664393681cb52b9a98b	1	1698362059035	0x3adf7250e143e03d1074e037f90c311b0a8b3281
0x3010c9c224e28f88f700994966145d7158bd9511	5	1703114860951	0xd5325dcf3ecbde108cec321c7bb78ae93c74a5ea
0x353defeafbcc3952897e2f7f9f2daefbbe12f4bc	11	1705996856123	0xabd75efb1aa2603ac1965d582131e82232da9aa3
0x3617dfccb7d25efcd246e9bde75b148cd60f1052	7	1692808227095	0x6f272fad2ed6f067979a6f5e963f5277f5d9908a


Ibukunoluwa Adisa,[1,2]
[1] Department of Phy
[2] Joint Quantum Institute,
[3] DEVCOM Army Rese
[4] Quantum Materials

tation of measurement outcomes correspo
(blue) and the excited states (orange) of
heterodyne readout, the measurement sig
both the power (red arrow) and the phase
measurement tone. (b) In self phase-refer
the signal is independent of phase drift.

prospect futures, LLC

bringing decentralized AI from the cloud to your operating system.

currently building a new enterprise SaaS tool to power the future of AI x Legal x Operating Systems

https://github.com/orgs/prospectfuturesinc/projects/3/views/1

Problem 2 Solution:
$\Sigma = \{0, 1\}$
$\Gamma = \{0, 1, A, B, \sqcup\}$

https://github.com/prospectfuturesinc/defense? 2. Hiring: https://arxiv.org/pdf/2507.07955 3.
https://www.youtube.com/watch?v=YGLNyHd2w10