SMART CONTRACT RISK ANALYSIS

~ Prakhyati Bansal

Webacy Externship

INTRODUCTION

- **Objective:** Provide insights into the frequency and correlation of smart contract vulnerabilities and propose actionable strategies to mitigate risks.
- **Scope:** The analysis covers multiple risk tags and explores their relationships using the Phi coefficient.

KEY VULNERABILITIES IDENTIFIED

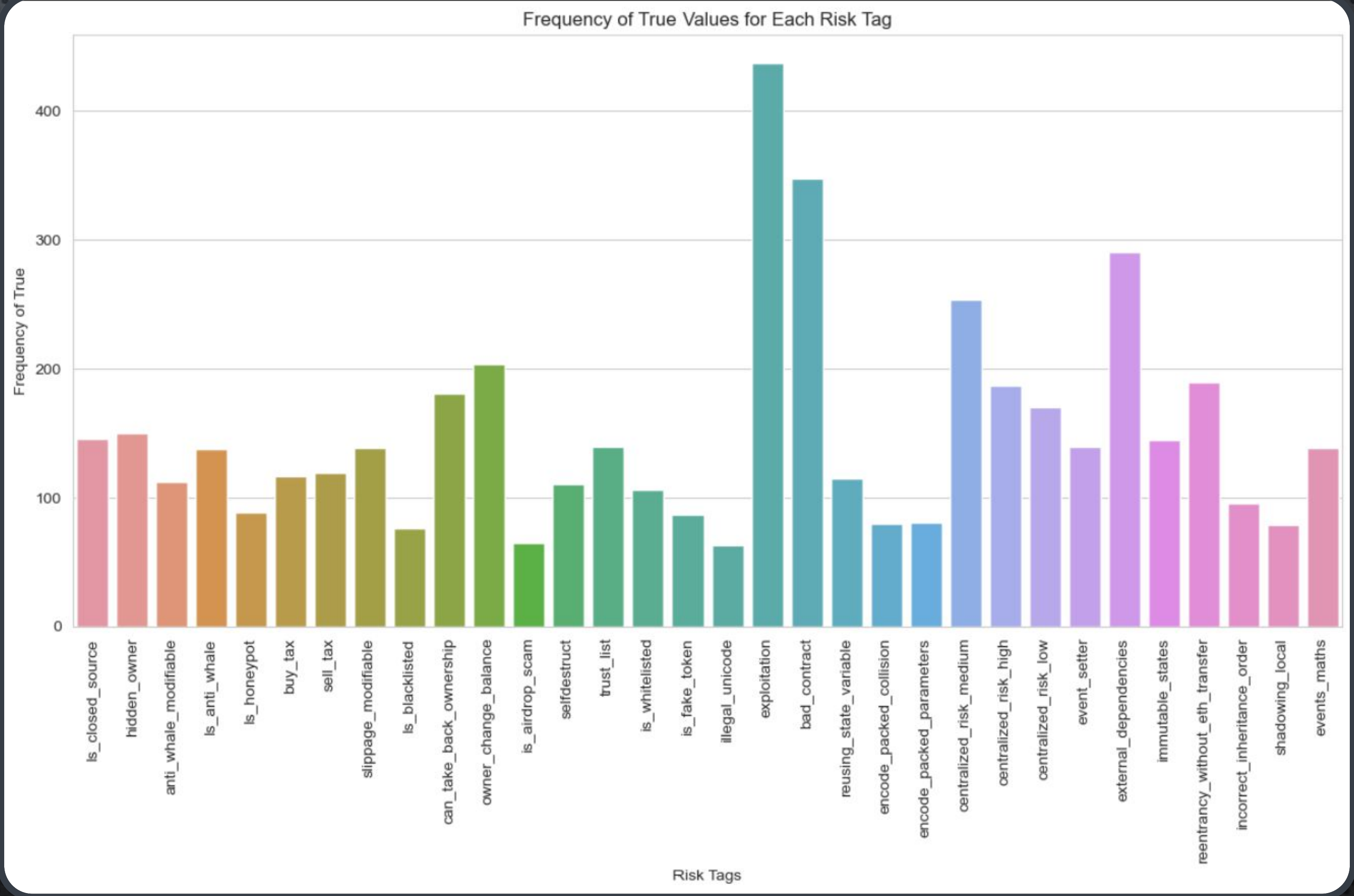
- MOST FREQUENT RISK TAGS:

1. EXPLOITATION

2. BAD CONTRACT

3. EXTERNAL DEPENDENCIES

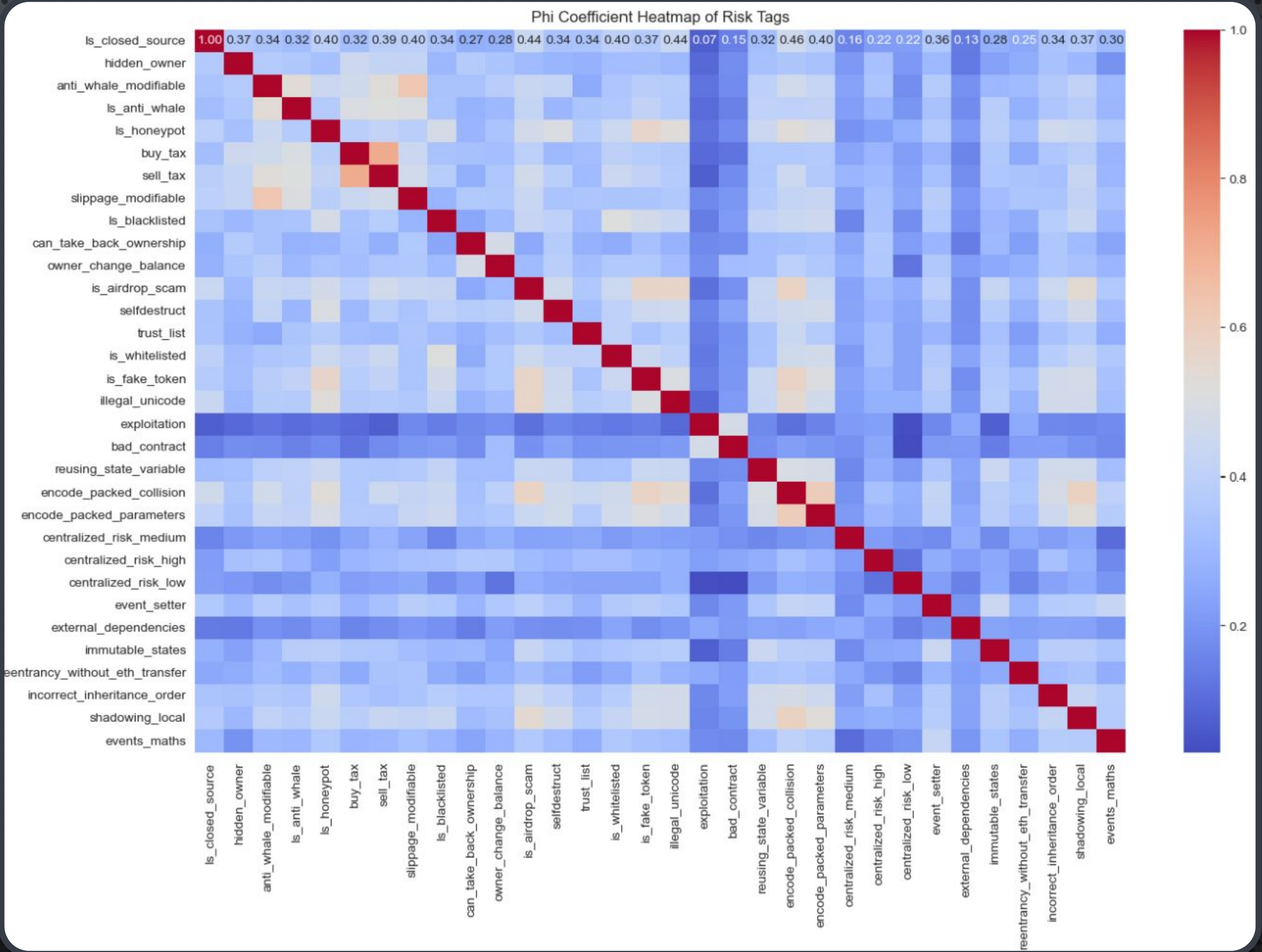
- THESE TAGS HIGHLIGHT AREAS OF HIGH VULNERABILITY IN SMART CONTRACTS, PARTICULARLY RELATING TO CODE QUALITY, RELIANCE ON THIRD-PARTY SERVICES, AND SUSCEPTIBILITY TO ATTACKS.



RISK TAG CORRELATION ANALYSIS

- Correlated Pairs:

1. **Buy Tax - Sell Tax:** These settings are often modified together, affecting user transaction costs.
2. **Slippage Modifiable - Anti-Whale Modifiable:** Indicates a relationship between preventing market manipulation by whales and liquidity management features.



STRATEGIC RECOMMENDATIONS

- **1. Strengthen Exploitation Prevention:**

  Action: Implement regular external audits and static analysis tools to identify exploitable vulnerabilities in contract code.

  Implementation: Automate code scanning tools during the development lifecycle to catch common issues early.

- **2. Address "Bad Contract" Issues:**

  Action: Establish a more robust internal review system and enforce code quality standards to avoid bad practices.

  Implementation: Use peer-review models and enforce coding guidelines across teams.

- **3. Reduce Risks from External Dependencies:**

  Action: Rely on trusted external contracts and perform comprehensive due diligence when integrating third-party contracts.

  Implementation: Isolate critical functions from external services and require thorough external contract audits before integration.

- **4. Monitor Transaction Fee Modifiability (Buy/Sell Tax):**

  Action: Set fixed thresholds for buy/sell tax and limit the ability of owners to modify transaction fees after contract deployment.

  Implementation: Use governance frameworks to manage transaction fee modifications transparently.

- **5. Manage Slippage and Anti-Whale Features:**

  Action: Conduct regular reviews of liquidity settings to ensure that the slippage and whale-prevention mechanisms are functioning as intended.

  Implementation: Implement automated alert systems to detect and prevent abusive changes to these settings.

REAL- WORLD IMPLEMENTATIONS

- **Blockchain Application Strategy**:

Smart Contract Development: Prioritize frequent audits and reviews, focusing on the high-risk areas highlighted (e.g., exploitation, external dependencies).

Governance: Incorporate transparent governance structures that restrict the ability to modify critical contract settings such as transaction fees or slippage.

Automated Security Checks: Leverage blockchain tools that monitor live contracts for unusual activity (e.g., slippage changes, large withdrawals) to detect malicious behaviors early.

Conclusion

- **Takeaways:**
  - Most frequent vulnerabilities ("exploitation," "bad contract") highlight a need for code quality control and security audits.
  - Correlated risks (buy/sell tax, slippage/anti-whale) require targeted oversight and restrictions on owner modifiability.
  - Strategic measures, including audits, automated checks, and governance models, can significantly improve the security and robustness of smart contracts in real-world applications.
- **Call to Action:** Incorporate these insights into future development practices to strengthen the security posture of blockchain ecosystems.