# Article information:

Potential Risks of Hyperledger Fabric Smart Contracts | IEEE Conference Publication | IEEE Xplore
<https://ieeexplore.ieee.org/abstract/document/8666486>

# Article summary:

1. Hyperledger Fabric utilizes general-purpose programming languages like Go, Node.js, and Java for smart contract development, which may introduce risks as these languages were not originally designed for writing smart contracts.

2. Existing tools in the Go community, such as golint and gosec, may not cover all potential risks associated with developing chaincodes in Hyperledger Fabric.

3. A prototype of a static analysis tool was developed to detect risks in smart contracts written in Go for Hyperledger Fabric, highlighting the importance of addressing vulnerabilities in permissioned blockchain frameworks.

# Article rating:

May be slightly imbalanced: The article presents the information in a generally reliable way, but there are minor points of consideration that could be explored further or claims that are not fully backed by appropriate evidence. Some perspectives may also be omitted, and you are encouraged to use the research topics section to explore the topic further.

# Article analysis:

The article titled "Potential Risks of Hyperledger Fabric Smart Contracts" provides a comprehensive overview of the risks associated with developing smart contracts on the Hyperledger Fabric blockchain framework. The article highlights the importance of ensuring the security and reliability of smart contracts, especially in permissioned blockchains like Hyperledger Fabric.

One potential bias in the article could be the focus solely on risks associated with smart contracts developed using general-purpose programming languages like Go, Node.js, and Java on Hyperledger Fabric. While it is important to address these risks, there may be other factors or vulnerabilities specific to other blockchain frameworks or languages that are not covered in the article.

The article does a good job of discussing existing tools and approaches for verifying and validating smart contracts, such as Oyente, Mythril, Securify, and ZEUS. However, it would have been beneficial to provide more detailed information on how these tools work and their limitations. Additionally, the article mentions the development of a new detection tool for covering risks not addressed by existing tools but does not provide enough information on how this tool works or its effectiveness.

Furthermore, while the article acknowledges some risks associated with non-deterministic chaincodes in Hyperledger Fabric, it does not delve deep into other potential risks or vulnerabilities that developers should consider. It would have been helpful to explore a wider range of risks and provide practical examples or case studies to illustrate these risks.

In terms of promotional content or partiality, the article seems to maintain a neutral tone and focuses primarily on presenting research findings rather than promoting any specific products or services. However, there could be a slight bias towards highlighting the advantages of using general-purpose programming languages for smart contract development on Hyperledger Fabric without fully addressing their limitations.

Overall, while the article provides valuable insights into potential risks associated with Hyperledger Fabric smart contracts and offers solutions through tool development, there are areas where more depth and exploration could enhance its credibility and usefulness for developers working in blockchain technology.

# Topics for further research:

* Risks of smart contracts on other blockchain frameworks
* Vulnerabilities specific to smart contracts in Ethereum
* Limitations of smart contract verification tools like Oyente and Mythril
* Non-deterministic chaincode risks in Hyperledger Fabric
* Practical examples of smart contract vulnerabilities in blockchain development
* Comparison of smart contract development in different programming languages on Hyperledger Fabric

# Report location:

<https://www.fullpicture.app/item/c2a2ce625341ea6bf2c5f71995e8e8c0>