

Alice in  
gnø.land

Miloš Živković



## What's on the menu

- 1.** Down the Rabbit Hole
- 2.** Through the Looking Glass
- 3.** Tea Party with the Mad Hatters



Miloš Živković

Core Go Engineer @ Gno.land



zivkovicmilos



@\_zmilos



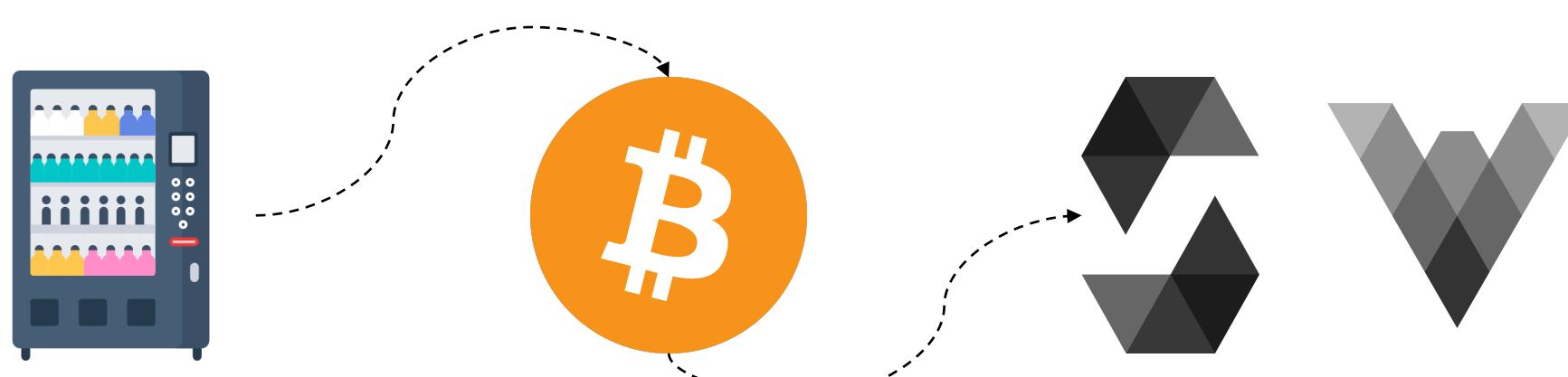
@zmilos

# Down the Rabbit Hole



## What are Smart Contracts, really?

“A smart contract is a computerized transaction protocol that executes the terms of a contract” – Nick Szabo (1994)



## These solutions...are not perfect

- The barrier of entry for newcomers is not minimal



### CRYPTOZOMBIES



### Solidity by Example

```
● ● ●

pragma solidity ^0.8.0;

contract ShortContract {
    struct Data {
        uint256 value;
        mapping(address => bool) addresses;
    }

    event ValueChanged(uint256 newValue);

    address private owner;
    mapping(uint256 => Data[]) private dataList;

    modifier onlyOwner() {
        require(msg.sender == owner);
        -
    }

    constructor() {
        owner = msg.sender;
    }

    function addData(uint256 index, uint256 value) external {
        dataList[index].push(Data(value));
    }

    function updateDataValue(uint256 index, uint256 dataIndex, uint256 newValue) external onlyOwner {
        Data storage data = dataList[index][dataIndex];
        data.value = newValue;
        emit ValueChanged(newValue);
    }
}
```

## These solutions...are not perfect

- Limited in design and purpose => Unlimited in exploits

Re-entrancy

Front running

Phishing with tx.origin

Arithmetic Overflow

Honeypots

Signature replay

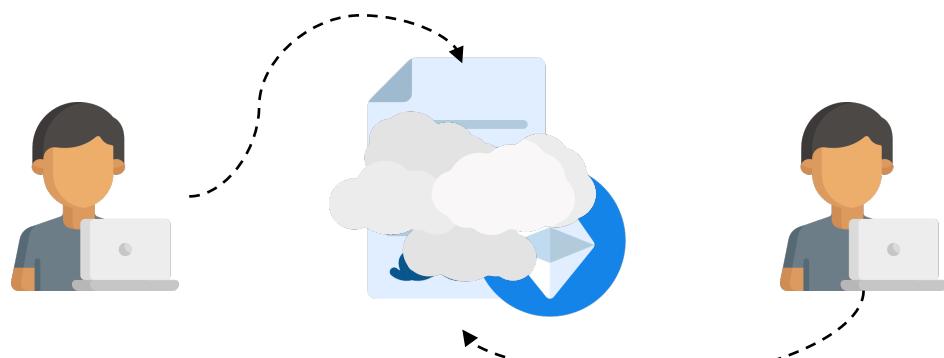
Arithmetic Underflow

Private data access

Denial of service

## These solutions...are not perfect

- No source code transparency and auditability



**Verify & Publish Contract Source Code**  
COMPILER TYPE AND VERSION SELECTION

 Source code verification provides **transparency** for users interacting with smart contracts. By uploading the source code, Etherscan will match the compiled code with that on the blockchain. Just like contracts, a "smart contract" should provide end users with more information on what they are "digitally signing" for and give users an opportunity to audit the code to independently verify that it actually does what it is supposed to do.

Please be informed that advanced settings (e.g. bytecodeHash: "none" or viaR: "true") can be accessed via Solidity (Standard-Json-Input) verification method. More information can be found under Solidity's "Compiler Input and Output JSON Description" documentation section.

Please enter the Contract Address you would like to verify

Please select Compiler Type  
[Please Select]

Please select Open Source License Type ⓘ  
[Please Select]

I agree to the [terms of service](#)

**Continue** **Reset**

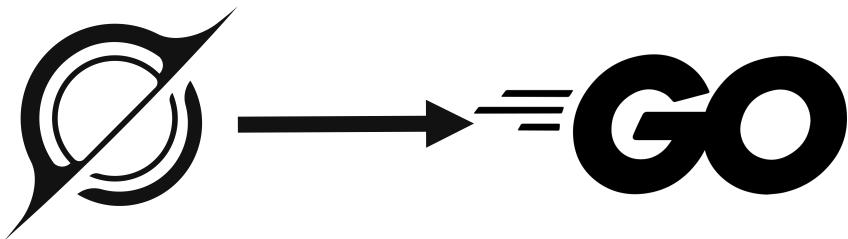
## What is Gno.land?

**Gno.land is a layer 1 smart contract platform that addresses key challenges in the blockchain space, with a focus on making smart contract development easy and intuitive for users**

**It leverages an interpreted version of Go, called Gnolang (Gno)**

## Gno is accessible

By basing the Smart Contract language on a version of Go, existing Web2 developers can easily transition into writing blockchain logic



```
● ● ●  
package hello  
  
func SayHello(name string) string {  
    return "Hello " + name + "!"  
}
```

## Gno is transparent

**Source code transparency is at the forefront of Gno.land, so users and developers can always know what they are executing**

**“Upload sources, not binaries”**

- @moul



## Gno is supercharged

**Gno.land gives you all the superpowers of Go, and more**

- **Simple, yet complete**
- **Designed for inter-chain SC systems**
- **Powerful “std” library, with a package marketplace**

# Through the Looking Glass



## Realms

**A Realm is a Smart Contract on Gno.land, written in Gno**

- **Contain state (stateful)**
- **Imported from gno.land/r/...**
- **Expose a Render method for Markdown**

## Packages

**A Package is simply a bundle of functionality**

- **Don't contain state (stateless)**
- **Imported from gno.land/p/...**
- **Can be imported by other Realms / Packages**

## What we're building

A ticketing system for...the Mad Hatters' tea parties!



date: ...  
time: ...  
guests: ...



date: ...  
time: ...  
guests: ...



date: ...  
time: ...  
guests: ...

- **RSVP as a guest**
- **See all available parties**
- **See all RSVP'd guests**

## Tea party with the Mad Hatters



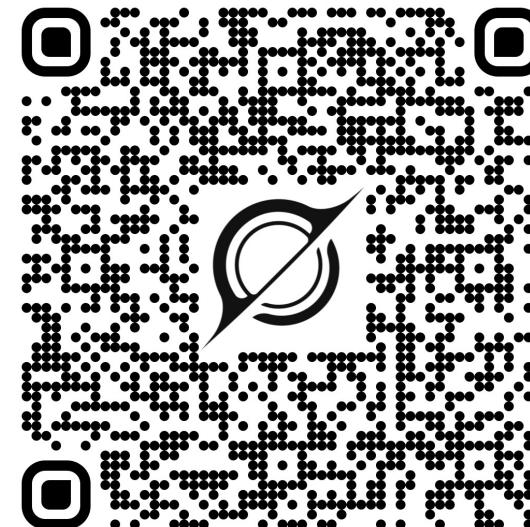
## What you'll need to get started

- Internet connection
- Git
- Go 1.19+

**GNO\_WIFI**  
**gno2023**

## Where can I find the tutorial?

You can find the entire workshop (text + code) on the following GitHub link:



Tea party with the Mad Hatters

# DEMO

How can I integrate Gno into my dApps?

Gno.land offers official package support for dApp developers

- Tendermint2 JS/TS client (SDK) – [tm2-js-client](#)
- Gno JS/TS client (SDK) – [gno-js-client](#)
- Golang client (SDK) – [Soon™](#)

What if I want to set up a wallet?

There are multiple wallets being implemented for Gno

- The Adena wallet by the Onbloc team is the most mature

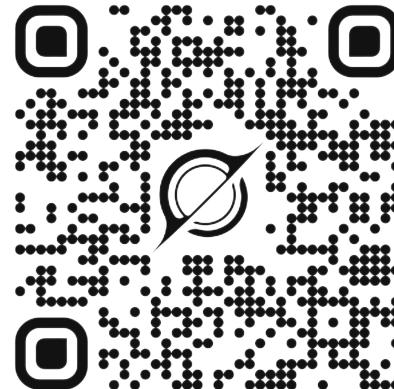
Where can I learn more?

**The GNO ecosystem is constantly improving...but you can make it better!**

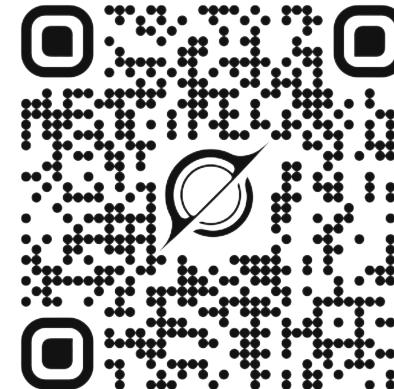
**GitHub**



**Awesome GNO**



**Discord**



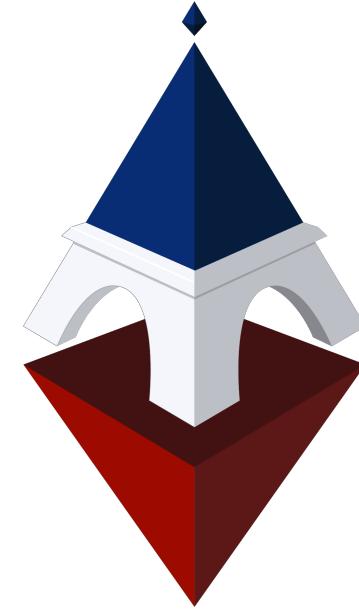
**Game of Realms is a worldwide developer competition challenging participants to build Smart Contracts and tooling in Gnolang.**

**Learn more on the official Discord:**



Tea party with the Mad Hatters

**See you at EthCC!**



gnø.land