**15. Are there protections or impediments that would prevent market participants or other actors from intentionally disrupting the normal function of the Ethereum Network in an attempt to distort or disrupt the Ether market?**

**Note:- This answer is the one from Github responses, as it is quite extensive and covers all the points we had researched.**

Ethereum currently secures billions of dollars in value using open-source code, so the security of the platform is an utmost concern to both developers and the general community. This vigilance has been warranted, given that the platform has experienced availability attacks, smart contract bugs, and ideological disagreements in governance. Through these challenges, the Ethereum platform has demonstrated that its multiple layers of protection can effectively secure value.

Ethereum’s defenses can be broadly categorized into four layers: well-written code, security-optimized system design, cautiously implemented crypto-economics, and effective governance.

Well-written code serves as the first line of defense. Ethereum developers have taken many steps to write secure code such as using battle-tested encryption methods, sharing secure design patterns, and implementing rigorous reviews and testing processes. Their efforts are not perfect, and mistakes often lead to significant consequences such as mistakenly locking up Ether forever or having Ether stolen due to security bugs. Even so, the uninterrupted operation of Ethereum since the Frontier launch in 2015 has shown that the system can remain operational despite these flaws.

Beneath the code, Ethereum’s security-optimized system design provides a second line of defense. The pieces of Ethereum work together in a way that minimizes the chance that a fault in one area will have systemic consequences. For example, the application layer is separated from the base protocol layer such that, if a smart contract proves faulty, it will not jeopardize the safety of the network. The well-publicized DAO attack is one great example. Even though an attacker had hacked a smart contract holding about a sixth of all available Ether, a sizeable minority argued that the platform did not need to make a protocol-level change to fix the issue. This position would not have been viable if the application layer was at all connected to protocol functionalities.

Another important system design decision is the encouragement of multiple clients with multiple versions that all sync to the same Ethereum blockchain. There have been numerous examples of how bugs in a blockchain’s client software can cause potentially critical failures in the system. Unlike Bitcoin, Ethereum has two major clients (geth and Parity) for its Proof-of-Work system that prevent bugs in one client from becoming systemic issues. The ecosystem’s security will improve further in this regard under Proof-of-Stake, where 8 new clients are under development.

After system design, a cautious implementation of crypto-economics serves as a third line of defense for Ethereum. Crypto-economics is defined as the usage of economics, cryptography, and game theory to design systems that generate predictable outcomes given certain assumptions. Ethereum’s code and system design use crypto-economics to ensure the network’s security. Bitcoin’s Proof-of-Work system pioneered this type of design (which Ethereum largely uses), while Ethereum’s new Proof-of-Stake system offers significant improvements. Proof-of-Work likely needs no introduction; in the 10 years since the introduction of Bitcoin, no one has successfully broken the system. Ethereum’s Proof-of-Stake improves on security by introducing new concepts such as staking and slashing, which increases the economic cost of attacking the network as compared to Proof-of-Work.

As explained elsewhere, making a switch in consensus system entails significant risks. A failure within the code or the system design might provide short or medium-term inconveniences to the network, but a failure of crypto-economics in the consensus system potentially threatens the network’s survival. To mitigate this risk, Ethereum researchers have designed a conservative, staged approach that migrates network activity to the new system without jeopardizing the old system. This approach minimizes the chances that flaws within the Proof-of-Stake system that may otherwise make the Ethereum platform unusable.

The final and most enduring source of resilience for Ethereum is its effective governance. Since the Ethereum yellow paper was first published in 2014, the community has grown from a small group of visionaries to a network of hundreds of thousands of technologists and enthusiasts around the world. During that time, it has only improved in its ability to act as effective custodians to the value secured by the network.

An example of the community’s growth comes from how it handled a broken smart contract written by Parity. On November 6, 2017, a bug in Parity’s wallet smart contract caused it to permanently lock away $300 million of community funds. While this bug did not affect as many people as the DAO hack, it directly impacted Parity, a development group which was responsible for the second biggest client that runs Ethereum. When Parity proposed a solution somewhat similar to how DAO hack was resolved (a fix submitted through a hard fork), the community quickly concluded that such a solution was not in the best long-term interests of the network. Right or wrong, the community showed the ability to decide quickly on a potentially controversial topic. As a result, this incident ended up causing little visible impact to the value of Ether.

Within the community, the Ethereum protocol developers have shown an exemplary capacity to put the long-term interests of the network ahead of short-term goals and personal ambitions. The recent decision to combine the development of Proof-of-Stake and Sharding illustrates this ongoing commitment. Originally, developers of Proof-of-Stake and Sharding had been working separately to design their respective solutions. Through the course of their research, it quickly became clear that a combined solution would be more secure and less complex. Of course, working together would come with the inevitable result of scrapping their existing work and delaying the timelines for release, but the developers quickly aligned on the new approach with minimal disagreements. Their willingness to change course exemplifies the community ethos to subordinate short-term rewards to long-term value-creation.

Ethereum has operated uninterrupted for three years as a decentralized blockchain. With continued improvements to coding practices, system design, crypto-economics, and governance, the community looks to keep the network running for decades to come.

**22. Are there any emerging best practices for monitoring the Ethereum Network and public blockchains more broadly?**

Due to the transparency of blockchains, there are multiple services that provide monitoring, visualization and analytical tools for Ethereum and related applications that are built on top. Monitoring on the Ethereum mainnet can be done by various ways. The most common way of monitoring the past transactions, block details, and the status of the network is through:

1. **Block Explorers**, most of which are web applications like – [Etherscan](https://etherscan.io/), [EtherChain](https://www.etherchain.org/), [Blockchair](https://blockchair.com/ethereum), and [BlockScout](https://blockscout.com/eth/mainnet). These block explorers are a group of tools which provide the users with the ability to view past transactions, details of the blocks mined and graphical representation of financial aspects related to the transactions. Things that can be explored with some of the block explorers are:

* Contract Addresses
* Non-Contract Addresses
* Transactions
* Blocks
* Contract Code

1. **DApps monitoring**, is another way in which the apps running on the mainnet can be monitored to get information about number of users as well as the contract addresses for that applications. [State of the DApps](https://www.stateofthedapps.com/rankings/platform/ethereum), [DappRadar](https://dappradar.com/rankings/protocol/ethereum), are some of the applications that give extensive list of dapps running on the Ethereum mainnet.
2. There also have been a group of developers who came up with their own ways of depicting the transaction details along with block details, in an attempt to help the users visualize these transactions and block creations. [Ethviewer](http://ethviewer.live/) and [crytolights](http://cryptolights.info/) are some of the attempts to depict Ethereum mainnet transactions and block creations in a visually appealing format.

There are various datasets available to download and analyze Ethereum data, collected by multiple organizations. Google [released dataset](https://www.kaggle.com/bigquery/ethereum-blockchain) from its BigQuery Public Data and a software system. The software system Google has built on its Cloud platform does several things: it synchronizes the Ethereum blockchain to computers running Parity; it pulls data from the Ethereum ledger on a daily basis, including the results of smart contract transactions; and it “de-normalizes and stores date-partitioned data to [BigQuery](https://cloud.google.com/bigquery" \t "_blank) for easy and cost-effective exploration”. Google also adds visualization for accounts that conduct transactions over the mainnet.

As each transaction by definition involves two or more parties, the Ethereum network can be seen as a giant graph of financial interactions. Several projects allow viewing parts of this graph by choosing account or contract of interest. [Observeth](https://observeth.net/) displays ether and token transfers in a given period of time. [Ethtective](https://www.ethtective.com/) focuses on showing accumulated graph of interactions for a given Ethereum account.

Firms like [Gauntlet](https://gauntlet.network/), are building a blockchain simulation and testing platform that leverages battle tested techniques from other industries to emulate interactions in crypto networks. Simulation provides transparency and greatly reduces the cost of experimentation so that teams can rapidly design, launch, and scale new decentralized systems.

Finally, there are application-specific visualization and analytic tools. Most of these tools are focused on financial protocols. For example, [MKR Tools](https://mkr.tools/) provides an overview of the debt positions made via Maker DAO. However, these Dapps can be monitored in depth by using DApps monitoring tools specified above.

**23. Are there security issues peculiar to the Ethereum Network or Ethereum- supported smart contracts that need to be addressed?**

Ethereum network relies mostly on deployment and execution of smart contracts which govern the transfer of Ether or ERC20 tokens over the network and that’s why securing the smart contracts becomes the most important aspect in securing the blockchain. However, smart contracts, like any piece of code, suffer from the possibility of having vulnerabilities or bugs. There have been multiple attacks which resulted from exploiting these bugs or unhandled exceptions in the smart contracts. The biggest impact on the Ethereum network was caused by **“TheDAO”** attack, followed by attacks like King of the Ether Throne (KotET) , GovernMental, EtherPot, SmartBillions and TheRun. The main reason for these attacks were unhandled exceptions in the smart contract code and the bugs which were exploited.

However, independent auditing and various tools preventing vulnerable smart contracts have been created.

* [ZeppelinOS](https://zeppelinos.org/) – “an open-source, distributed platform of tools and services on top of the EVM to develop and manage smart contract applications securely". Zeppelin introduces a novel approach in developing smart contracts by using already developed and secure smart contracts (i.e. libraries). Doing so, presumably will lead to mitigating severe vulnerabilities which are related to programming mistakes. Furthermore, the off-chain component provides numerous tools like debugging, testing, deployment and monitoring
* [SolCover](https://github.com/sc-forks/solidity-coverage) – a tool that measures and describes the degree of overall testing in a smart contract. Even though, it does not serve as a mechanism to identify specific vulnerabilities, it could be argued that it creates a more secure environment with the philosophy that more tests = more secure.

Security audits are considered to be the most effective way of identifying vulnerabilities in a pre-deployment phase. Experienced blockchain developers, and specialized teams, carefully investigate the smart contract manually and automatically to identify possible vulnerabilities, and make sure that it follows best programming practices. Despite the fact that it might be the most secure method for preventing deployment of vulnerable smart contracts, it is not considered to be popular because of the high price range that security audit firms have.

**24. Are there any best practices for the construction and security of Ethereum wallets, including, but not limited to, the number of keys required to sign a transaction and how access to the keys should be segregated?**

**Note:- This answer is the one from Github responses, as it points out all the possible explanations.**

There is a wide range of solutions tackling the problem of storing and managing Ethereum assets a.k.a. creating and using an Ethereum wallet. Solutions vary in the security, usability, and dependence on third-parties. All wallet products can be divided into two groups depending on who owns and controls the private key. First, there are centralized solutions that own the private keys and therefore the underlying assets. They can potentially censor user's actions and are vulnerable to hacks as they become honeypots, but can offer features like account recovery, shorter passphrases, and overall better user experience. Second, there are self-custody solutions where a user is in control of her private key, which is more secure and reliable, but at the same time, a user carries a risk of permanently losing her funds if she'll ever lose the private key. It's worth to mention that there're projects like [Gnosis Safe](https://safe.gnosis.io/) aiming to deliver secure yet convenient and reliable wallet software where a user can be sure that funds will be safe even in the event of losing the private key without relying on a single service provider.

Speaking about self-custody Ethereum wallets, there are implementations of multisignature wallets that are audited and battle-tested. Probably the biggest one in terms of adoption is [MultiSigWallet](https://github.com/Gnosis/MultiSigWallet). Another solution is [Simple Multisig](https://github.com/christianlundkvist/simple-multisig). To get some insight about what is the industry average in terms of the number of required keys to sign a transaction, one can look at the existing deployments of MultiSigWallet used by various Ethereum projects. For example, [Aragon's multisig](https://etherscan.io/address/0xcafe1a77e84698c83ca8931f54a755176ef75f2c) is 2-of-3, [Bancor's wallet](https://etherscan.io/address/0x5894110995b8c8401bd38262ba0c8ee41d4e4658) is 2-of-4, and [Golem's contract](https://etherscan.io/address/0x7da82c7ab4771ff031b66538d2fb9b0b047f6cf9#code) is a 3-of-N multisig.

In terms of creating new smart contracts for safe custody and management of assets, one can look for the best practices in the ecosystem. Consensys [Ethereum Smart Contract Best Practices](https://consensys.github.io/smart-contract-best-practices) provide advice on what direction to follow when writing a contract, as well as highlight various caveats. [ETHSecurity](https://www.ethsecurity.org/) serves as a curated list of everything related to writing secure contracts including blogs, lectures, and tooling.Finally, there are tools designed to find potential vulnerabilities and bugs in the smart contract source code. To name a few, there are [Mythril](https://github.com/ConsenSys/mythril-classic), [Manticore](https://github.com/trailofbits/manticore), and [Echidna](https://github.com/trailofbits/echidna).

**25. Are there any best practices for conducting an independent audit of Ether deposits?**

By the virtue of blockchain technology, everything on the public blockchains is recorded in the public ledger which makes it open for anyone, who is connected to the blockchain, to verify balances of any account or smart contract without any interruptions. Alternatively, one can use a block explorer like [Etherscan](https://etherscan.io/) or [Blockscout](https://blockscout.com/), which involves some trust in the honesty of the service provider but doesn't require from the user to run a node.

Many firms and organizations provide blockchain auditing services to provide track and trace for transactions from deposit/source to destination. Additionally, private sector CPA firms including PwC, Deloitte, EY, and Accenture now provide full-service forensics for cryptocurrencies and blockchains. However, a private blockchain, which does not reveal sensitive information about the transacting accounts, would need a different approach for auditing. There's a lot of research ([Aztec](https://www.aztecprotocol.com/), [Enigma](https://enigma.co/)) directed towards general-purpose cryptographic schemes that will enable private transactions on Ethereum. In most cases, to audit the balance of a private account or smart contract, one would need to cooperate with the owner of such account in some way. The exact way of collaboration will largely depend on the privacy solution that will be used. One example might involve sharing "view keys" with the auditor, which will allow viewing the balance of the wallet, but not to move the funds. Another solution is to use zero-knowledge proofs that can reveal some properties of the underlying wallet (say, there are more than 10,000 ETH in that wallet) without revealing the exact amount of funds.