DEBUGGING THE EXPENSIVE WAY

Blockchain users are rediscovering hard lessons from real-time systems and game theory. By **Chris Edwards**

"I accidentally killed it," a programmer with the handle devops199 confessed on a Github message board. But unlike most situations where a developer had accidentally taken down a service or deleted a critical function, this particular error destroyed $300m of deposited cash.

Since the error in November 2017, the money remains locked away within the Ethereum blockchain. It can only be recovered by a "hard fork" in which users are forced to a new version of the chain, a move for which it is hard to find a consensus among users partly because it calls into question the claimed immutability of blockchain records.

The code killed by devops199 was a set of library functions for a so-called smart contract. Blockchain advocates use the term smart contract to describe small programs that run on blockchain networks and which manage transactions between users.

Such contracts are not so much smart as simply ways to automate record-keeping processes traditionally performed manually under conventional legal "dumb" contracts.

In an industry such as electronics manufacture, a smart contract might be invoked every time a shipment of components changes hands on its way to the factory. Although this could be handled by supplier databases, a blockchain implementation makes it easier for multiple parties to cooperate and open the door to easier trading between contract manufacturers.

One with a surplus of a particular batch of ICs could advertise them by placing them on the blockchain. Another manufacturer facing a shortage can bid for and buy those parts through one part of the smart contract and then assign a courier to pick them up, at which point the contract updates the record to show who has custody.

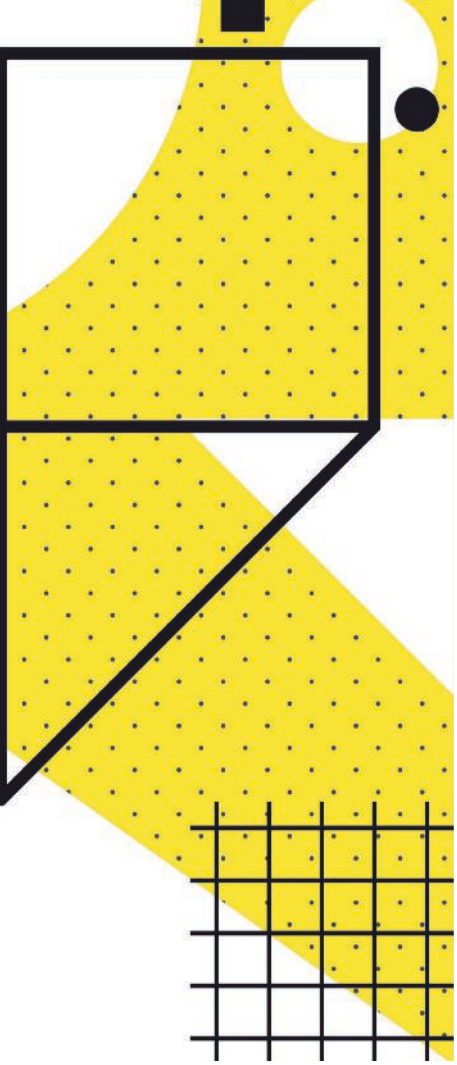
Most smart contracts in use today control wallets that store cryptocurrency tokens, such as Ethereum's Ether, making it easier to move the money around and enforce controls on how transactions proceed. For example, a company can provide multiple users with access to the store of funds and impose transfer limits using smart contracts.

A key aspect of smart contracts is that everyone can see and interact with the contents of every smart contract posted to the blockchain. It is up to the smart-contract writer to enforce controls to prevent this work from interfering with their operation. For some reason, devops199 went about their research into smart contracts they found on the Ethereum network in a peculiarly destructive way on code that had not been properly protected. "I'm eth newbie…just learning…sending kill(), destroy() to random contracts. You can see my history," the programmer wrote in a chat room later, ending with a sad emoticon.

"Can't make an omelette without breaking some eggs, I guess," wrote another chat-room occupant.

Most of the attempts by devops199 had apparently failed. The kill requests were simply rebuffed by the code inside the smart contracts

Ivan Kamzyst / Alamy Stock Vector



or had little practical effect. But one attempt succeeded in spectacular and expensive fashion. Because of a small but critical error made by Parity's developers in writing the software, a package of library functions were not declared as such. Instead, they appeared as smart contracts in their own right. This gave the code an unfortunate property: it made it possible for a developer to claim ownership of it.

Once they had ownership of the contract users could do what they liked, such as kill it. And kill it they did, along with the money it controlled. Such problems are not limited to blockchain networks.

The blockchain concept

The 'open-source' software world has itself faced the consequences of a developer killing off creations. Appalled by a trademark-infringement over the naming of one of his modules on a popular open-source Javascript repository, developer Azer Koçulu removed all of his code including a function that padded the left side of strings with zeroes. This function had been used in numerous other libraries that suddenly refused to build.

An important difference is that the left-pad function code could be easily restored to the repository, albeit in a legally questionable way. The key difference with the blockchain is its guarantee of irrevocability. On top of this is the way in which blockchain development is, today, intertwined with finance. Because they suck in real money by the million, accident-prone blockchains are guaranteed to make headlines.

David Wong, security consultant at NCC Group, argues one way to look at a network such as Ethereum as simply being "a very big computer". It is also a very slow computer. This is a consequence of the way in which all actions need to be recorded through the consensus-forming process that has emerged as one of the main features of blockchain systems since the creation in January 2009 of the Bitcoin protocol.

The blockchain concept described in the Bitcoin white paper written by its creator under the pseudonym Satoshi Nakamoto borrows ideas from several sources. The most obvious was a timestamping system for official documents devised in 1991 by Stuart Haber and W Scott Stornetta, who worked at Bellcore. Their proposal was to create a chain of cryptographically signed blocks, with each successive block's signature dependent on its predecessors. In the original proposal, the timestamps used to show authenticity would be created by recording data from the latest block in the form of a cryptographic hash and publishing it on Usenet or in a newspaper.

Nakamoto's protocol replaced the need for publication off-chain with a consensus algorithm that could be used in a peer-to-peer system. In this proof-of-work system, "miners" collect the data for the next block in the chain and generate a hash that must have a certain number of leading zeroes by adding in a "nonce" value. As the output of a hash function is almost impossible to predict without performing the function itself, it can take a large number of attempts to come up with a hash with the required format. The first miner to succeed is paid. If two miners happen to publish a result simultaneously, the successful recipient is the one whose version of the blockchain propagates more widely and receives additional the blocks afterwards.

"Unlike most situations where a developer had accidentally taken down a service or deleted a critical function, this particular error destroyed $300m of deposited cash."

The idea of Ethereum and similar networks being large decentralised computers is fundamental to making blockchain-based system able to execute smart contracts and not just exist as stores of notional cash value: the primary use of Bitcoin, the most successful blockchain so far.

Because of the time and computational effort needed to complete each transaction, blockchain processing is considerably slower than trusting the changes to a third party with a dedicated computer. For applications that need a combination of transparency, permanent records of transactions and an unwillingness to rely on a single provider to maintain the records, the blockchain has distinct advantages. But with those properties come attributes that have tripped up numerous developers.

Many of the issues encountered in blockchain programming will seem familiar to those used to programming real-time systems, with concepts such as race conditions and re-entrancy faults as well as even simpler programming mistakes such as not dealing with arithmetic underflow and overflow. Although the consensus algorithms provide a strong guarantee that transactions were made, they make no attempt to check whether they should have happened. The result has been that those discovering problems the hard way cause million-dollar losses rather than a board locking up or a small motor burning out.

The first major loss due to a core

mistake in the smart contracts running on a blockchain led to the collapse of the DAO crowdfunding network. A hacker took advantage of a function that allowed re-entrancy without any checks to withdraw money time and again before account balances were updated. Once the blockchain updated, the wallet behind the victim's smart contract would be empty. The attacker obtained the equivalent of $60m although Ethereum agreed to allow a hard fork to reclaim a large proportion of the money.

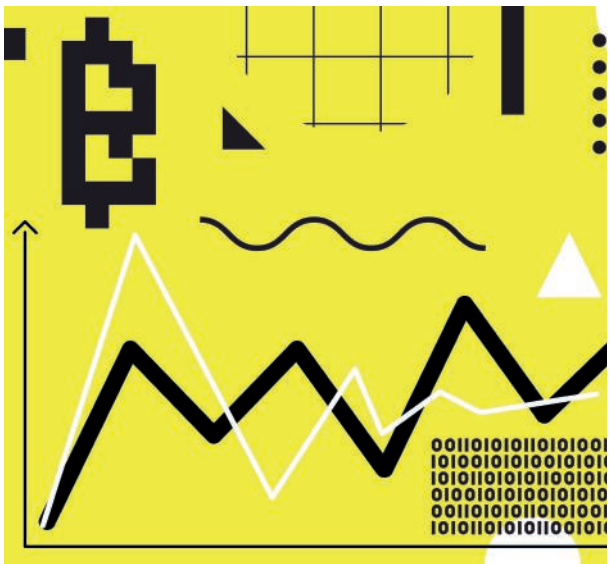
Whereas real-time programmers have to face the vagaries of interrupts and their influence on the timing of processes inside their systems, at least conditions such as race conditions can be written off as the consequences of random chance. Blockchains also suffer from the often malign influence of game theory.

Security assumptions

"Some of the security assumptions we have to make are economic," Philip Daian, a postgraduate researcher at Cornell University, explained at Ethereum's Devcon4 in Prague late last year.

For example, the assumption behind the consensus mechanism that underpins Bitcoin is that the miners who control the blockchain will not subvert it because to do so would undermine the value of the tokens traded on the network. A 2018 paper by Eric Budish, professor of economics at the Chicago Booth School of Business, argued sabotage is another motive for corrupting a blockchain though a miner has to spend heavily in a proof-of-work system to achieve enough control.

In the case of Bitcoin, only nation states are likely to have the clout to perform a majority attack, limiting their probability. Alternative blockchains, particularly if they suffer a decline in usage, would be far more vulnerable although researchers from the Digital Currency Initiative at MIT have argued the practical barriers to a successful



majority may be higher than Budish has calculated. But a concentration of power is not just possible it seems likely for blockchain systems. Analysis 18 months ago by researchers at Cornell University found more than 90 per cent of the mining power on Bitcoin was owned by eleven mining pools on Bitcoin and just seven on Ethereum.

More subtle attacks on the integrity of blockchains come through race conditions, another problem from real-time computing that has resurfaced in the world of consensus protocols though with some additional wrinkles from the world of economics. For example, because all transactions are public, front-running afflicts Ethereum in much the same it does stock and futures exchanges.

Front-runners spot potentially lucrative transactions that they can gazump. They step in to satisfy them with a fee that is more attractive to miners and so wind up winning the order. On blockchains, this is not necessary as they can simply take advantage of the mining latency. Miners may also choose to reorder competing transactions in ways that are more advantageous to them.

To avoid these kinds of attack, writers of smart contracts will need to agree on locked prices rather than submit to spot pricing or find techniques, such as so-called submarine sends, to hide details until mining has completed for the block that contains them.

"In the case of Bitcoin, only nation states are likely to have the clout to perform a majority attack, limiting their probability."

Although it has taken some time for the industry to react to its challenges, blockchain developers now recommend more stringent programming practices that take transaction security into account and looking at ways in which technology might help.

Wong says because smart contracts are usually small in terms of lines of code they are amenable to techniques such as formal verification. And formal verification projects focused on blockchains have sprung up. At Devcon4, Professor Mooly Sagiv of Tel Aviv University and co-founder of Certora said his company expects to launch a freemium service next month that will check smart contracts for consistency. The tool would test for generic properties that all contracts should obey, such as the ability to perform combinations of transactions atomically, and against specific use-cases described using the company's own Contract Vulnerability Language.

This second part is where Certora is trying to build a portfolio of specifications to cover common applications such as asset ownership and money-market trading systems. "The interesting thing about this language is that it's reusable: we have the ability to write reusable specifications," says Sagiv. But it will take time to develop the necessary specifications.

"We need a community effort. We need to understand what you are doing so we can understand the correctness requirements."

Until the tools are in place to assess the quality of blockchains and their smart contracts, users who want to embrace the technology will still need to tread carefully. And make sure they have learned the lessons of programming history.