Post-mortem: 0x v2.0 Exchange Vulnerability

0x Core Team
Jul 18, 2019 · 11 min read

What happened?

On July 12, we were made aware of a potential exploit in the 0x v2.0 Exchange contract by samczsun, a third-party security researcher. The vulnerability would allow an attacker to fill certain orders with invalid signatures. After verifying the vulnerability internally, the 0x Core Team triggered an emergency shutdown mechanism within 0x protocol's system of smart contracts, preventing the vulnerability from being exploited. Historical trade logs confirm that the vulnerability had not been exploited prior to its discovery and disclosure.

Subsequently, a patched version of the smart contract system (0x v2.1) was deployed to the Ethereum mainnet. The initial announcement may be found here.

> *It is important to note that no one exploited this vulnerability, no user funds were lost, and the ZRX token contract was unaffected.*

What is the technical explanation for the vulnerability?

0x v2.0 introduced the Wallet signature type, which allows smart contracts that hold a user's assets to sign orders on behalf of the user. The discovered vulnerability took

advantage of an error in the Wallet signature verification logic that would allow a malicious actor to create valid orders on behalf of any externally owned account (EOA) that had approved the 0x contracts to spend their assets. Let's take a look at the code in question:

```
assembly {
    let cdStart := add(calldata, 32)
    let success := staticcall(
        gas,              // forward all gas
        walletAddress,    // address of Wallet contract
        cdStart,          // pointer to start of input
        mload(calldata),  // length of input
        cdStart,          // write output over input
        32                // output size is 32 bytes
    )

    switch success
    case 0 {
        // Revert with `Error("WALLET_ERROR")`
        mstore(0,
0x08c379a0000000000000000000000000000000000000000000000000000000)
        mstore(32,
0x0000002000000000000000000000000000000000000000000000000000000000)
        mstore(64,
0x0000000c57414c4c45545f4552524f5200000000000000000000000000000000)
        mstore(96, 0)
        revert(0, 100)
    }
    case 1 {
        // Signature is valid if call did not revert and returned
true
        isValid := mload(cdStart)
    }
}
return isValid;
```

This code made some incorrect assumptions about the inner workings of the EVM that lead to this critical vulnerability:

- Calling a function on an EOA would result in an exception (instead, it *always* returns a "successful" status).
- Specifying a return data length in a contract call would always result in that amount of bytes being written to memory (in reality, a call that returns less than the length specified will only write the actual return data length to memory).

Because these 2 assumptions failed, an `isValidSignature` call to an EOA would always be treated as successful. Since the output was supposed to overwrite an area of memory that was already non-zero, the return data would also always be treated as `true`.

For more detail on this vulnerability, see the posts written by Samczsun and ConsenSys Diligence.

Why did the 0x Core Team write this particular function in assembly?

The vulnerable function was implemented in assembly because Solidity did not support the ability to make a staticcall at the time the contract was written. Ironically, the Exchange v2.0 contract used the staticcall opcode in order to *increase* the safety of the contracts. This opcode is used to guarantee that a call cannot update state, which prevents many attack vectors by limiting the possible functionality of external contracts. The staticcall opcode was introduced in Ethereum's Byzantium network upgrade and was not yet available in Solidity at the time the vulnerable 0x v2.0 Exchange contract was deployed. Amir and Remco provide more context on the decision on Twitter.

Learn more about staticcall here.

What will the 0x Core Team do to ensure that this doesn't happen again?

- Take advantage of recent Solidity features that reduce the 0x pipeline's reliance on assembly workarounds. Many new features such as `abi.encode`, `abi.decode`, and low level calls returning data allow the use of assembly to be minimized in the 0x contracts. In fact, this exact vulnerability could not be exploited in the 0x v3.0 contracts that are currently in development - one of the first areas that were changed was converting this particular piece of code to Solidity and checking the length of the actual data returned.

- Do not over-optimize code by using assembly. Small efficiency improvements are not a concern in the current environment. The side effects of security flaws, on the other hand, can have compounding effects that bubble up to applications and protocols that make use of 0x contracts. Security is the single most important consideration when writing smart contracts.

- Add strict rules and guidelines around the usage of assembly within the codebase. All values should be properly validated by masking bits and checking lengths. External calls to contracts that support interfaces used within the protocol should expect a magic value to be returned and check that the return value matches this *exactly*. Memory management should be handled with extreme caution.

- Stay on top of Solidity development. While Solidity adds many safety checks to code under the hood, it is always possible that Solidity itself contains bugs that could result in exploits.

- Upgrade our suite of testing and security tools. We will make static analysis, fuzz testing, and combinatorial testing a standard part of our testing process, in addition to 100% test coverage through unit and integration tests. A good litmus test for the addition of these tools will be to see if they could have caught this particular vulnerability.

- Explore formal verification of the 0x smart contracts. While this will not provide full safety guarantees for the entirety of the 0x contracts, we can prove that certain properties and state transitions will always hold.

How did the migration and recovery process play out?

- **4:30 PM PT:** Samczsun reached out to engineers at 0x and explained the potential vulnerability.

- **5:20 PM PT:** Test cases were written that took advantage of and verified the exploit.

- **7:45 PM PT:** Emergency shutdown of the 0x v2.0 Exchange contract. Authorizations to the ERC20Proxy, ERC721Proxy, and the MultiAssetProxy were removed.

- **7:56 PM PT:** Stakeholders in the 0x ecosystem notified of an emergency shutdown.

- **9:38 PM PT:** 0x v2.1 patch completely written and tested.

- **9:43 PM PT:** Analysis of historical trade logs complete, confirming that the vulnerability was never exploited on mainnet.

- **10:21 PM PT:** 0x v2.1 contracts deployed to mainnet.

Within 24 hours of the vulnerability's disclosure, TokenLon, Radar Relay, Paradex, Bamboo Relay, and LedgerDex completed the migration to 0x protocol v2.1 along with a number of large liquidity providers.

Open orderbook liquidity recovered soon after trading resumed. See, e.g., slippage for $10K trades for ETH-DAI on Radar Relay:

Thank you to our peers in the community for the outpouring of support and understanding.

Are other Ethereum-based projects vulnerable to this exploit?

Solidity added support for static calls in v0.5.0, so Solidity developers no longer need to use assembly to perform static function calls. Thus, the vulnerability is unlikely to appear in contracts written after the Solidity v0.5.0 update.

The 0x team tested for this vulnerability in a handful of other mainnet contracts that use similar signature schemes. While the exploit does not appear to affect any of the contracts we studied, we were not able to conduct an exhaustive search of all deployed contracts. It remains possible that vulnerable contracts exist in the wild. We suggest that Ethereum smart contract developers take time to learn about the exploit and perform their own investigations.

Why did neither the 0x Core Team nor independent security audits catch this?

As this situation has demonstrated, one can't guarantee with 100% certainty that a code base is free of bugs, even after it has passed multiple independent security audits. In this case, a seemingly simple snippet of assembly — which was needed to perform a static function call — baked in assumptions about how the EVM behaves.

Is Ethereum's VM too complex for safety-critical contracts?

Ethereum development is similar to other security-critical engineering efforts (aerospace, nuclear engineering, industrial control, etc.) and should be approached using similar quality assurance techniques. The EVM is well-understood and ahead of alternatives in its ability to support complex decentralized systems with high assurance. Like any developing technology, it has known shortcomings.

Will samczsun receive a bug bounty?

Yes, we run an ongoing bug bounty program. Due to the seriousness of the vulnerability discovered by samczsun, they will receive a full $100,000 reward. The bug bounty

program remains active and now covers the 0x v2.1 smart contracts deployed to mainnet.

What other permissions do the 0x contracts currently provide to the 0x Core Team?

The 0x Core Team controls the AssetProxyOwner contract, which can plug in or remove smart contracts from the pipeline **following a mandatory 2 week timelock**, that is binding on-chain. This speed bump gives the community ample time to observe and react to any pending change. In the future, the AssetProxyOwner contract will be controlled by an increasingly sophisticated and decentralized binding on-chain governance system.

Why wasn't there a two week grace period before the 0x v2.1 contracts went live?

0x protocol's system of smart contracts typically relies on a "hot upgrade" process that is similar to a soft fork in that it does not require users to reauthorize the 0x smart contracts (users do have to formally approve of the upgrade via token vote for it to take place). Normally, it takes 2 weeks for an upgrade to go into effect after being approved and initiated on-chain, which gives users time to exit the system if desired.

In the interest of preventing network downtime after the vulnerability was discovered and patched, the 0x v2.0 smart contracts were completely shut down and a new system of 0x v2.1 smart contracts were deployed from scratch. As a result, traders will need to provide the patched set of smart contracts with access to their digital assets. The transition to 0x v2.1 is analogous to a hard fork of 0x protocol and the associated liquidity network, since it requires users to proactively migrate to the new forked system.

Does the 0x Core Team believe that decentralization is important?

Yes. Bitcoin's censorship-resistant design inspired many of us to enter the crypto space and we are deeply committed to contributing to the development of an open, globally accessible financial system. We believe that basic financial services and tools should be available to everyone in the age of the internet, regardless of the nation-state in which one is born. Since the beginning, our goal for 0x protocol has been for it to serve as a public good that is governed by a global online community. Removing points of

centralization from 0x protocol such that the 0x community has full sovereignty is core to our mission.

The decision to put guard rails in the 0x smart contracts does not reflect our philosophy on decentralization. Today, we are faced with the realities of developing safety-critical financial infrastructure on top of a rapidly evolving tech stack and we must balance our passion for decentralization with what we believe to be common sense security backstops.

Ultimately, the entire 0x protocol codebase is open source and freely available under the Apache 2.0 license. Stakeholders within the ecosystem determine the canonical set of 0x smart contracts by choosing to use them. However, the network is only useful if relayers, liquidity providers, developers, and traders work together to make it so.

Finally, decentralization exists on a spectrum and there are a number of ways it can be defined. Balaji proposes the Minimum Nakamoto Coefficient as a metric for quantifying a crypto network's level of decentralization. In the case of governance over a system of smart contracts, the level of centralization for the entire system is equal to that of the *most centralized subsystem*. The existence of an emergency shut off mechanism within the 0x protocol contracts yields an undesirable minimum Nakamoto coefficient because it is controlled by only a few private keys rather than hundreds of thousands of them. However, our goal has always been to transition binding control over to ZRX stakeholders incrementally over time.

What is the timeline for moving to a binding on-chain governance system?

It is important to note that while the current governance process is not technically binding on-chain, honoring the ZEIP process and the will of the community is at the core of our values. We take this responsibility extremely seriously and will always do our best to *do the right thing* and *focus on long-term impact*.

A prerequisite for any *binding* on-chain governance mechanism is incentive alignment between the class of users that bring value to the platform and those that govern it. If the governance mechanism relies on token voting, then incentive alignment means that tokens are staked by and/or delegated to the class of users that drive value to the platform and those that view it as a public good. The easiest way to create incentive alignment between the two groups is to ensure they consist of the same people. In the

case of 0x protocol, the group that provides the platform with value are the liquidity providers.

Putting this all together, we need to see liquidity providers controlling a significant percentage of voting power before it makes sense to make governance binding on-chain. If approved in early Q4, the incentives proposed in ZEIP-31 should begin to shift the token distribution and voting power towards active liquidity providers.

How could a DAO promptly mitigate issues like this if tokenholders must vote on all changes?

A number of people raised the point that a DAO would not be able to rapidly respond to a security incident in the same way that our team was able to because (1) coordinating tokenholders is time-consuming and (2) the vulnerability would need to be disclosed to tokenholders before they could be convinced to use the emergency shut down mechanism, but disclosing the vulnerability to tokenholders is equivalent to full-blown public disclosure.

A reasonable compromise might be for tokenholders to elect a small technical committee whose members control the emergency shut down mechanism. The committee would need to be nimble, the members deeply familiar with Solidity, the 0x contract architecture and tech stack, but also sufficiently distributed so no government could target a quorum of committee members simultaneously. The committee would serve as the first point of contact for the 0x bug bounty program, respond to vulnerabilities, and payout rewards to hackers that disclose vulnerabilities.

How will the 0x DAO or decentralized governance system be structured?

The term “DAO“ has become somewhat of a catch-all term used to describe a broad spectrum of governance systems implemented within smart contracts. According to Vitalik’s terminology guide, it is more accurate to describe the governance system we envision for 0x as a decentralized organization (DO) because the system will control a pool of capital (community treasury), rely on humans at the edges of the network (makers, takers, relayers, developers, tokenholders), and social consensus will play an important role in directing the protocol’s upgrade path and the allocation of community resources.

In short, staked ZRX tokens (see ZEIP-31) will be eligible to cast votes on technical upgrades and to elect representatives onto committees that serve the ecosystem. Peter Zeitz's presentation from Devcon4, "Building self-sustaining ecosystems through governance", describes a committee-based approach to funding ecosystem development. This will be revisited once ZRX staking has been live long enough for liquidity providers to have significant voting power. It is important to note that this represents our current understanding of the best path forward. There are a number of Ethereum projects experimenting with innovative governance systems that we are closely following and that we hope to learn from. And, ultimately, the 0x governance system will be able to evolve with the needs of the community.

Ethereum Crypto 0xproject Cryptocurrency Bitcoin

About Help Legal