Transcoder Slashing Vulnerability Retrospective

Protocol

---

dob #1 July 30, 2019, 4:57pm

On Thursday, July 29th the Livepeer security team received a report of a possible protocol issue discovered by security researcher **samczsun**. After confirming that this bug was in fact present in the protocol, was of a critical severity that would have put user funds at risk, the team took the steps to pause the protocol, prepare and test a patch, and deploy and verify the update and restart the protocol within an approximate 9 hour period. **This issue was not exploited and no user funds have been compromised**.

These sorts of issues are expected during the alpha, will likely occur again, and they are the exact reason that the protocol is upgradable at this early phase. This post aims to provide transparency around the process used to discover, assess, and fix the issue, so that everyone can participate in, and improve upon this process going forward.

The Issue
Samczsun has done **a great writeup** with full details including the code. We will summarize here for the community as well.

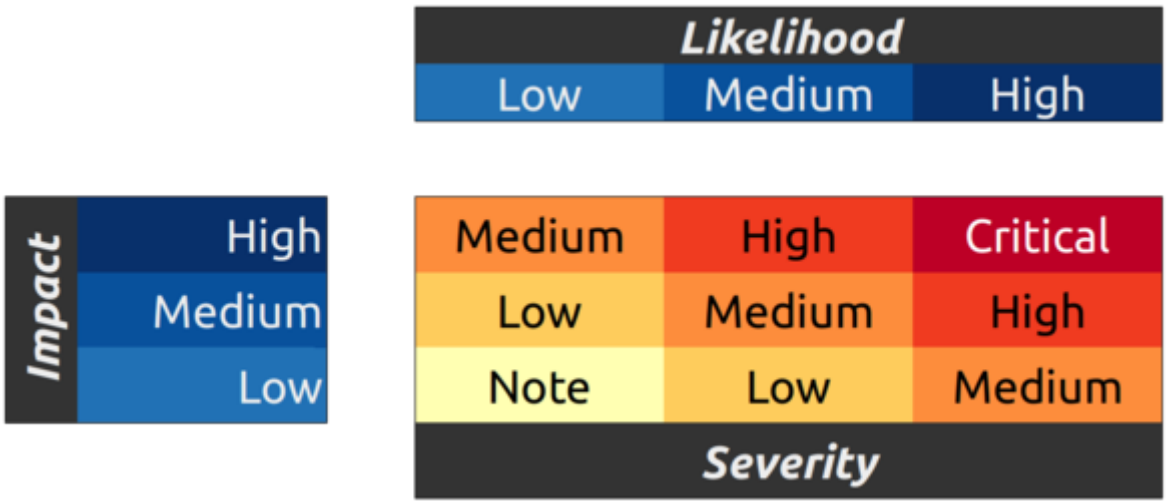
In Livepeer, Transcoders perform video transcoding jobs for broadcasters. Jobs consist of encoding many multi-second segments of video. At the completion of a job, they submit `claimWork()` transactions to the protocol specifying how many segments they encoded, and then they get paid accordingly.

If the transcoder tries to “Double Claim”, or essentially report that they encoded the same segment multiple times in order to be paid double for the same work, users can detect this and submit a `doubleClaimSegmentSlash()` transaction. If that function determined the transcoder did in fact double claim, the transcoder would be slashed 3% of their stake and deactivated from the set.

The bug that was discovered was that this function was missing an important validation on the input arguments, failing to check if the same job and claimIds were passed in as the inputs, therefore making it possible to slash transcoders in invalid cases.

The Effect

According to Livepeer’s threat matrix, this is classified as a critical severity issue, as it has both a high probability of occurance, and a high impact, with potential loss of transcoder staked funds.



- This would have resulted in improper slashing of transcoders
- This would have yielded a “findersFee” for the user who invoked the slash transactions, hence creating an incentive to use this exploit.

The Fix
The fix was a **one line code change** to add a `require()` statement to ensure that the same claim can not be used as both inputs to the slashing transaction. This was tested via unit tests, integration tests, on Rinkeby testnet, and verified on mainnet.

The Process
Upon discovering this issue, the Livepeer security team entered into the following process:

- Identify top priorities - protect protocol user value and trust in the protocol.
- Analyze the issue to understand it.
- Assess the various options for addressing the issue, including deciding whether to pause the protocol.
- Decided that due to user value being at risk, pausing the protocol to protect funds was warranted.
- **Communicate the issue publicly**, and steps for resolution - assuming a 4 hour fix window.
- Create an incident response checklist.
- Review and test the proposed fix - in this case a one line update to the protocol smart contracts - tested on Rinkeby test network and with unit and integration tests.
- Deploy the fix
- Verify
- Communicate the update and fixed protocol - occurred in just over 4 hours from the protocol pause and 9 hours from the initial report.

The Timeline

The following events all took place on July 29, Eastern Daylight Time.

- 11:29am - Report received to the **security@livepeer.org** disclosure email address
- 12:20pm - Livepeer team begins internal investigation according to our security incident response process above
- 4:21pm - Protocol is paused. Issue is communicated publicly.
- 8:40pm - Fix has been deployed and verified and protocol is unpaused.

Bug Bounty
Samczsun will be receiving a $5000 bug bounty for his responsible disclosure of this critical severity

issue.

Lessons learned

Unlike some of the previous issues that have required fixes during the alpha phase of the Livepeer protocol, this bug was not due to some subtle Ethereum complexity, or some deeply complicated logic within the inner workings of the Livepeer protocol accounting. This was a hiding-in-plain-sight logic error, that could have been caught at many stages including: development, code review, unit testing, integration testing, protocol audit, testnet, or the 1.5 years of mainnet usage of the protocol. However, as previously noted, it is important to recognize from the beginning that in a highly complex piece of software like Livepeer, there will always be bugs, and it's important to have the right procedures and mechanisms in place to deal with them an recover from them. Here are some of the other lessons that this incident re-inforced:

- Bug bounty programs and responsible disclosure policies are good things to have in place to encourage proper reporting, rather than exploitation, of issues.
- Security response procedures are good things to have in place to allow you to respond quickly without having to fly by the seat of your pants in the moment. We have practiced this at Livepeer and run through a similar process for multiple issues over the years.
- Ability to protect user funds is important during the early stage of a network - the mechanism to pause the protocol isn't used lightly, and it's a challenge to find the tradeoff between full decentralization and ability to protect users during an alpha and bug discovery. More to come on this in the coming weeks as we explore the next steps in the path to decentralization, but it is worth noting that 0x proposed a nice potential solution recently, which is that token holders could vote to enable a committee that has the ability to respond quickly to incidents like this.
- Code reviews and audits won't find all the bugs. You can never unit test your value-moving smart contract functions enough, especially when it comes to the full range of input conditions.

Thanks to the Livepeer community for the support and patience during the protocol pause and the issue resolution. And thanks to the team members who worked hard to get this fixed without missing a protocol round.

*(Note: samczsun is the same researcher who identified the **0x exchange vulnerability** last week, and there are many similarities in the response and resolution process between the two issues. Nice work Sam - the Ethereum ecosystem owes you a debt of gratitude.)*

2 Likes