Debugging verified external contracts with Truffle Debugger

Let's say you're crafting a clever flashloan transaction that interacts with five different DeFi dapps -- but on trying it, it doesn't work as expected. You open the transaction in Truffle Debugger to get a closer look at what went wrong. But how will you debug a transaction that the debugger doesn't have the source code for?

Well, new in Truffle release v5.1.29 is the debugger's--fetch-external flag! This new flag tells the debugger to searchEtherscan's repository of verified contracts and automatically download and compile those sources. As long as all the contracts you're interacting with are verified on Etherscan, you're good to go!

This means you'll be able to debug Mainnet / Görli / Ropsten / etc. transactions that interact with external contractswithout the need to have external contract source code available locally !

Now, before you get too excited, some caveats apply: - This only works with Solidity contracts, since @truffle/debugger only supports Solidity at this time. - While using this feature, you won't be able to load and unload transactions in the debugger like normal; you'll have to quit and restart to change transactions. - In order to be able to use the debugger and have it find the right source to download for your network, you'll need to connect to an archive node that offers thedebug_traceTransaction JSON RPC method. (See Geth'sManagement APIs).

That last one is a pretty big caveat, butthere's good news here ! Our sibling SaaS projectTruffle Teams , now lets yousimulate live networks via the free Sandbox feature. This works without requiring you to run your own node or to pay for someone else to run their own node, since Truffle Teams offers 1 free sandbox per GitHub organization!

Of course, you're free to use the debugger with whatever compatible Ethereum node you like, including command-line Ganache; but here's how to use Truffle Teams, which offers so much more, for this feature.

Example: Debug ENS in a [mostly] empty Truffle project

Click to enlarge

This screenshot showstruffle debug stepping through a Görli transaction to the ENS registry. Once you upgrade to Truffle v5.1.29, you can recreate this by doing the following:

1. Find theGörli ENS contract page on Etherscan
2. .
3. Find a recent transaction hash. (Seeexample from screenshot
4. .)
5. Sign up / login to Truffle Teams
6. and make a free Sandbox that forks off Görli.Note
7. : At the moment, you'll have to make sure that you manually specify that the Sandbox should use the same network ID as the chain it's forked from (in this case, 5). In the future, however, this step will be handled automatically. You can read more about Sandboxes inour earlier blog post on the subject
8. .
9. Copy your sandbox JSON RPC url to the clipboard.
10. Runtruffle init
11. and add this to your newtruffle-config.js
12. (replacing""
13. with your clipboard contents from step 4):
14. module
15. .
16. exports
17. =
18. {
19. */ ... rest of truffle-config.js .../*
20. networks
21. :
22. {
23. */ ... rest of networks .../*
24. sandbox
25. :
26. {
27. url
28. :
29. ""
30. ,
31. network_id

32. :
33. 5
34. }
35. }
36. }
37. 5a.Optional
38. . If you have anEtherscan API key
39. or feel like making one, you can specify that as well in yourtruffle-config.js
40. to speed up downloads:module
41. .
42. exports
43. =
44. {
45. / *... rest of truffle-config.js ...*/
46. etherscan
47. :
48. {
49. // replace this with your API key
50. apiKey
51. :
52. "0123456789abcdef0123456789abcdef"
53. }
54. }
55. Copy the transaction hash from step 2 and run the following (replacing
56. with the0x
57. -prefixed transaction hash):
58. truffle debug --network sandbox --fetch-external

And that's it! You're now debugging an ENS transaction in an empty Truffle project. You can step through and set breakpoints; inspect both decoded variables and the raw binary state; and get return values, revert messages, and stacktraces just like normal.

Now this example is a minimal one just to show how things work. But it will work just as well in your existing projects on transactions that feature your contracts in your project interacting with external verified contracts deployed on live networks.

Right now "verified contracts" just means contracts verified on Etherscan, but we're also looking at supporting the newSourcify project, which provides a more decentralized alternative to Etherscan's repository. We plan to add this in an upcoming release. But as of right now, you can use this feature with Etherscan's large library of verified contracts!

Update : As of Truffle v5.1.32,Sourcify is now supported! Truffle Debugger will now check bothEtherscan andSourcify for verified contract sources!

Hope you find this as exciting as we do! Thanks for using Truffle!