Homework 3: Breaking Smart Contracts

Patrick McCorry

Kings College London, UK
patrick.mccorry@kcl.ac.uk

Abstract. We'll focus on how to build and break self-enforcing smart contracts. Remember, they are typically around 40 lines of code, but they hold millions of dollars of asserts. It is recommended to make notes on this homework sheet for future use.

1 Why do we care?

In 1997, Nick Szabo published a blog about the idea of a *smart contract* where contractual clauses can be embedded in software. He envisioned a future where *all sorts of property that is valuable* will be controlled by digital means:

> *Smart contracts reference that property in a dynamic, proactively enforced form, and provide much better observation and verification where proactive measures must fall short. And where the vending machine, like electronic mail, implements an asynchronous protocol between the vending company and the customer, some smart contracts entail multiple synchronous steps between two or more parties.* - Nick Szabo

For awhile (and sometimes still today), it is often claimed that lawyers and coders will work together to write smart contracts. While this may be the case for some applications, it is in fact a restrictive view to think that smart contracts only deal with legal issues. As we have learnt in class, smart contracts are publicly verifiable and programmatic third parties who solve the *co-ordination problem*. They oversee (and self-enforce) a protocol's execution whenever there is no appointable third party:

> To be clear, at this point I quite regret adopting the term *smart contracts*. I should have called them something more boring and technical, perhaps something like *persistent scripts*. - Vitalik Buterin[1]

Over the years, we have witnessed the deployment of high-profile smart contracts holding millions of dollars of assets including the Ethereum Name Service (160k+ eth), WrappedEther (2.1m+ eth) and DigixDAO_MultiSig (395k+ eth). At the same time, we have also witnessed pseudonymous attackers exploit bugs in contracts to drain (or freeze) their coins including TheDAO (3.6m+ eth) and

[1] https://bitcoinist.com/vitalik-buterin-ethereum-regret-smart-contracts/

Parity's multi-signature contract (514k eth). Before you can build smart contracts in the wild, you must learn how to break them.

In this homework, we'll focus on general information about smart contracts before deep-diving into some attacks. The coding aspect will be covered during the lab session, and instead we'll focus on the theory.

2 Understanding smart contracts

Let's focus on the background information about smart contracts (and their capabilities).

- What is a smart contract? [2 marks]
- Name and explain two types of accounts on Ethereum. [4 marks]
- Name and explain three types of transactions in Ethereum [6 marks]
- What is gas? And how do we set its price? [4 marks]
- At a high level, what is the purpose of Solidity code, operation codes and bytecode? [3 marks]

2.1 Breaking smart contracts

As we have mentioned, pseudonymous attackers (think devops199) have taken down (and drained) smart contracts holding millions of dollars of assets. Let's consider some of the classic examples.

- What is a front-running attack for smart contracts and how can an average user do it without collaborating with a miner? Please explain with an example. [6 marks]
- What is the checks-interaction-effects paradigm? Please provide an example. [3 marks]
- Why is a contract-in-the-middle attack possible if `tx.origin` is used instead of `msg.sender`? [4 marks]
- What is an out-of-gas exception and how can it be used to lock up coins? [4 marks]

```
contract NotAPonzi {

  function playGame() public payable {
    require(msg.value== 1 ether); // 1 coin deposit

    // Contract has 10 eth?
    if(address(this).balance == 10 ether) {
      msg.sender.transfer(address(this).balance);
    }
  }
}
```

Fig. 1: A broken contract

– Figure 1 outlines a contract that is vulnerable. How can a malice contract use `selfdestruct` to attack it? [4 marks]
– At a high level, how does a re-entrancy attack work? And how can we prevent it? [6 marks]

3 Advanced Contracts

We hope you enjoyed the attack section! By the end of this class, you'll be a master of adversarial thinking! Let's take a break from attacks, and focus on some advanced smart contract features.

– What is an oracle smart contract? And why do we need it? [4 marks]
– What is a smart contract factory? And why is it useful to let another smart contract verify the code deployed? [4 marks]
– How can we configure smart contracts to support updates in the future? [4 marks]
– Name and explain three permission systems that can be built into smart contracts. [6 marks]

While the answers will be released in a week's time, if you found any of the questions difficult then please visit Patrick McCorry during his office hours.