Guld code review

Introduction

As of April 5th, 2018, The Guld organization[1] hosts 21 publicly accessible source code repositories on GitHub ("The Codebase"). The purpose of these repositories is to implement concepts described in the document "Guld: Universal Protocol for Relative Consensus"[2] ("The Whitepaper"). **This review shall evaluate the following claims:**

1. **The Whitepaper describes a technically novel system that will benefit its users in ways existing tools cannot.**

2. **The Codebase is a reasonably mature implementation of The Whitepaper benefitting from 2 years of development.**

Summary of Findings

The Whitepaper describes a way of using existing software to solve a problem that has already been solved. It does not give a precise description of how this solution should be implemented. **We find it unlikely that the system described in The Whitepaper will benefit its users in any novel way.**

The Codebase does not represent a mature implementation. The volume of code is consistent with less than one week of a single developer's work. The Codebase is not a meaningful technical contribution.

Claim 1

The Whitepaper begins with the claim that existing blockchain software categorically fails to solve problems faced by businesses and governments. This claim is foundational to The Whitepaper because it is used to argue that Guld will solve problems not yet addressed by existing blockchain software. The claim is supported by three citations:

> "[existing blockchain software] has so far been insufficient to address the problem of the Firm(3), let alone the Government(4). This restricts blockchains to the realm of prices and payment systems, and outside of the firm.(5)"

We expect citation 3 to support the claim *existing blockchain software is insufficient to address problems faced by businesses.* However, the citation links to a retrospective analysis of the DAO, a particular system of improperly implemented smart contracts whose

[1] https://github.com/guldcoin
[2] https://guld.io/docs/guld-whitepaper.pdf

use and subsequent failure led to some negative publicity for Ethereum. It is not appropriate to generalize from a single failure the notion that Ethereum cannot express the kinds of smart contracts required by businesses. Therefore, the cited document does not support the claim above. On the contrary, the authors expect “smart contract security will increases *[sic]* over time through experience.”

We expect citation 4 to support the claim *existing blockchain software is insufficient to address problems faced by government*. The citation links to the bitcoin wiki article on the block size controversy. We could interpret the citation to imply that existing software cannot operate at the scales government applications would demand, e.g. fiat currency, but software currently exists that achieves distributed consensus on millions of transactions per second,[3] sufficient for any fiat application. There are many government applications[4] that do not require high scalability, and that are unrelated to the cited article. Again, the cited document fails to support the claim.

We expect citation 5 to support the claim *blockchains are restricted to the realm of prices and payment systems, outside the firm*. Cited is the article “The Nature of the Firm” by R. H. Coase, published in 1937. This article argues that hiring employees to form a firm can have economic advantage over contracting out work at market price. Thus, it supports the implication *if blockchains are used only to drive markets, then blockchains are not useful in the firm.* This citation does not support the first part of the claim: *blockchains are restricted to the realm of prices and payment systems.* This claim can be refuted with a quick google search.[5] Thus, the cited document fails to support any part of the claim.

The Whitepaper fails to provide evidence to support the claim that Guld will solve problems not yet addressed by existing tools.

Throughout The Whitepaper, technical terms are introduced without definition in contexts where the definition is not clear: “symmetrical”, “relativistic”, “perspective”, “polytree”. The examples put forward and questions begged in the narrative are extended into definitions and rules for a consensus system. Little justification is provided for these design choices. The given definitions lack the precision needed for them to be useful. Aside from the examples and the invalid citations discussed above, no arguments for the correctness or usefulness of this system are put forward.

---

[3] https://www.algorand.com/ implemented, benchmarked: https://people.csail.mit.edu/nickolai/papers/gilad-algorand-eprint.pdf

[4] https://www.forbes.com/sites/forbestechcouncil/2018/01/25/developments-and-adoption-of-blockchain-in-the-u-s-federal-government/

[5] https://www.economist.com/sites/default/files/plymouth.pdf

For example, the paper defines a **Proof of Labor (Observation)** as the following:

1. [Participant X] signs & publishes unique thought A, along with public key [XKey]
2. Observer validates signature of [XKey]
3. Observer decides on uniqueness of thought A as proof of a living perspective
4. If Observer accepts thought A as unique, Observer can safely infer that the signer [Participant X] using key [XKey] is also alive

A definition is not given for “unique thought.” A procedure is not given for deciding on the uniqueness of a thought. At a glance, “unique thought” is a vague and technically unhelpful concept.

The Whitepaper suggests particular technologies should be used to implement the protocol: git, GnuPG, Ledger, OpenSSH, gitolite. Details of how these technologies should be used are vague, impractical, or insecure. For example, the paper suggests Guld users should “establish P2P socket connections with their friends.” For each user to maintain such a set of connections by hand is pointless toil. In most blockchain implementations, the process of peer discovery and connection is automated.[6]

Barring the specific directory structure, **the protocol described in The Whitepaper is already in common use:** Millions of software developers regularly use git in a distributed manner. The observations consumed and produced are software. Some developers sign commits and patches with GnuPG.[7] Several systems already exists for establishing and maintaining the relative trust of PGP public keys and the associated identities.[8]

Overall, The Whitepaper is more of a philosophical journey than a technical document. It builds a way of using existing software to solve a problem that has already been solved on a foundation of imprecise and unhelpful definitions. The Whitepaper describes a system that is not technically novel. Users of this system will not benefit in any way not easily achieved using existing tools.

Claim 2

Review of python-guldlib

We evaluated the python-guldlib repository at commit 8b2b096f [9], which was current master as of April 5th, 2018. We present some findings characteristic of the whole codebase in order of importance:

1. python-guldlib substitutes function inputs directly into shell commands. For instance, username is substituted into the shell command built on line 67. This is a command

[6] https://bitcoin.stackexchange.com/questions/3536/how-do-bitcoin-clients-find-each-other
[7] https://git-scm.com/book/en/v2/Git-Tools-Signing-Your-Work
[8] https://en.wikipedia.org/wiki/Web_of_trust and https://en.wikipedia.org/wiki/Public_key_infrastructure
[9] We have preserved a copy of python-guldlib for reference.

injection vulnerability[10] and leads to arbitrary code execution if an attacker can control the content of username. Even if this vulnerability isn't exploitable through tools that currently use python-guldlib, other programmers may assume that get_balance works properly on unsanitized input, leading to exploitable code. This is a "fail deadly" implementation.

2. Guld uses GNU Privacy Guard[11] (aka. GnuPG, gpg) to sign and verify data (eg. copies of a ledger) against PGP private keys[12]. The guld code interacts with gpg via the python-gnupg[13] module. This is not the recommended or standard choice for interacting with gpg from python code; the way in which python-gnupg enables interaction with gpg can expose guld to unnecessary risk of fault and compromise. The gpg authors recommend using GPGME to interface with gpg from python and other languages.[14]

3. The guld ledger implementation[15] is riddled with bad practice and potentially insecure code[16].

    a. Obvious duplicated code, lack of appropriate abstraction

        i. Lines 52, 59, 67, 69, etc. involve construction of subdirectories with a common prefix. This should be factored out into a function.

        ii. The functions get_guld_sub_bals, get_guld_overview, get_assets_liabs, and get_balance all construct similar shell commands. These similarities should be factored out.

        iii. The functions get_assets_liabs and get_balance do almost the same thing and repeat most of each other's code. These should be combined into one function with a parameter to select the different behaviour.

    b. Interacting with the shell

        i. Guld balances are retrieved by executing a shell pipeline consisting of the tools grep, find, some shell intrinsics, and Ledger.[17] This design is fragile because grep, find, and the shell intrinsics behave differently in

---

[10] https://nets.ec/Command_Injection
[11] https://www.gnupg.org/
[12] For an overview of the service GnuPG provides, see https://georgebrock.github.io/talks/pretty-good-introduction/
[13] https://pypi.python.org/pypi/gnupg
[14] https://wiki.gnupg.org/APIs and https://pypi.python.org/pypi/gnupg
[15] https://github.com/guldcoin/python-guldlib/blob/8b2b096f294c763849341c7d384252ae9868454d/guldlib.py
[16] These are "code smells" https://blog.codinghorror.com/code-smells/ - these are widely recognized to indicate poor code quality and to correlate with presence of bugs and vulnerabilities.
[17] https://www.ledger-cli.org/

different environments (eg. flavour of Linux, user's configuration, shell choice, system language settings).

ii. The tools used to implement Guld are intended for manual use; automated use by python-guldlib may surface race conditions.

iii. The shell pipelines built by python-guldlib may fail in dozens of different ways. These failure cases are not handled.

c. The functions gen_redeem_registration_fee and gen_register perform arithmetic on asset balances represented as floating point numbers. The results are formatted in a way that implies they shall later be fed back into the ledger. Floating point numbers do not represent values precisely and can, over time, lead to large amounts of the asset being "lost" to rounding errors.[18]

4. The code in this repository is concerned with retrieving ledger balances. It does not implement ledger updates or any of the trust operations described in The Whitepaper.

The remaining repositories that contain Guld-related code have an average of 5 commits each. In total, there are 1,760 lines of Guld-related application code across all 21 repositories. This indicates an insignificant amount of work overall: **we believe a single developer familiar with the tools involved could reproduce this work in 2 - 5 days.** Those repositories offering any significant code to review have problems similar to python-guldlib.

The code in these repositories does not represent a mature implementation, and is not a meaningful technical contribution. Most of the ideas described in The Whitepaper are not implemented here.

[18] https://stackoverflow.com/questions/3730019/why-not-use-double-or-float-to-represent-currency/