Contract and Protocol Validation/Verification

September 25, 2019

guha.jayachandran@sjsu.edu

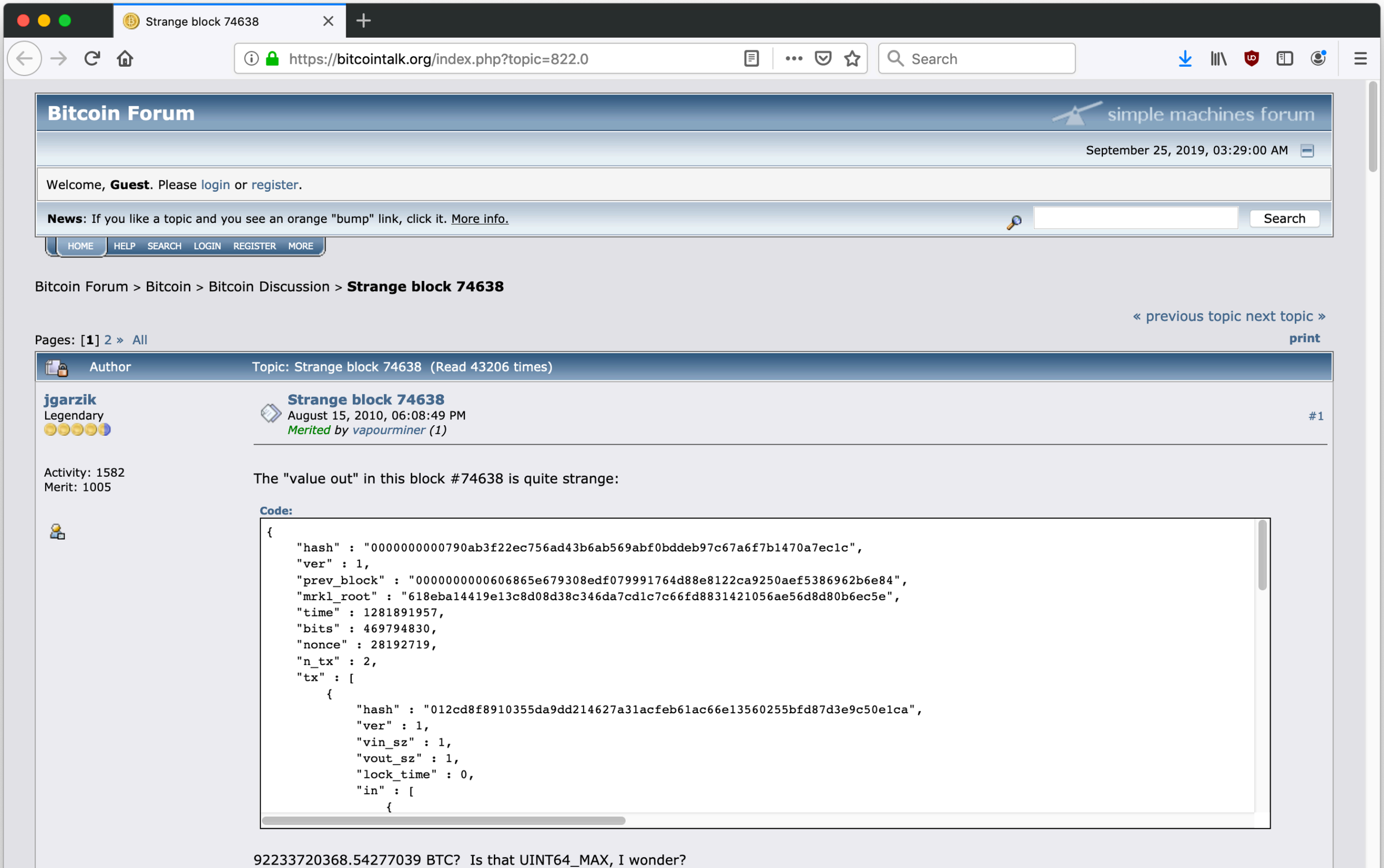
Announcements

- Only submit a question/takeaway slip at the start of class
- Homework due next Wednesday:
  - Submit hard copy unless otherwise arranged, 1 copy
  - Monday before/during class is your final chance to ask questions—no email after class Monday
- Final project information on last slide

Monday, we talked about there being many buggy smart contracts.

Even protocol code and protocol designs have had many flaws!

BTC Block 74638



- 2010
- Bitcoin had already been 4 major bugs discovered and fixed
- This was an inflation bug: the creation of coins out of thin air
- 184 billion new BTC
- Exploitation of the bug was noticed within 90 minutes
- Satoshi and Gavin Andresen fixed it within 2 hours
  - Why was speed important?

I applied your changes manually, and uploaded the result here: http://yyz.us/bitcoin/patch.bitcoin-gavin-overflow-quick-fix

Jeff Garzik, Bloq CEO, former bitcoin core dev team; opinions are my own.
Visit bloq.com / metronome.io
Donations / tip jar: 1BrufViLKnSWtuWGkryPsKsxonV2NQ7Tcj

satoshi
Founder
Sr. Member

Activity: 364
Merit: 2168

Re: overflow bug SERIOUS
August 15, 2010, 08:59:09 PM

#6

Here's the preliminary change. Look right? I have more changes to make, this isn't all of it. Will SVN shortly.

Code:

```
bool CheckTransaction() const
{
    // Basic checks that don't depend on any context
    if (vin.empty() || vout.empty())
        return error("CTransaction::CheckTransaction() : vin or vout empty");

    // Check for negative and overflow values
    int64 nTotal = 0;
    foreach(const CTxOut& txout, vout)
    {
        if (txout.nValue < 0)
            return error("CTransaction::CheckTransaction() : txout.nValue negative");
        if (txout.nValue > 21000000 * COIN)
            return error("CTransaction::CheckTransaction() : txout.nValue too high");
        nTotal += txout.nValue;
        if (nTotal > 21000000 * COIN)
            return error("CTransaction::CheckTransaction() : txout total too high");
    }
```

Don't sticky the topic, nobody looks up there. There'll be enough posts to bump.

satoshi
Founder
Sr. Member

Activity: 364
Merit: 2168

Re: overflow bug SERIOUS
August 15, 2010, 09:06:45 PM

#7

It would help if people stop generating. We will probably need to re-do a branch around the current one, and the less you generate the faster that will be.

A first patch will be in SVN rev 132. It's not uploaded yet. I'm pushing some other misc changes out of the way first, then I'll upload the patch for this.

kencausey
Newbie

Re: overflow bug SERIOUS
August 15, 2010, 09:09:53 PM

#8

lfm
Full Member

Activity: 196
Merit: 100

Re: Strange block 74638
August 15, 2010, 07:34:18 PM

#6

Im speculating here somewhat but from what I can see someone has generated a transaction, probably using a custom modification of the software to generate a transaction which exploits a weakness in the code. The code check each transaction output for negative numbers individually (up to ver 0.3.8 at least) but forgot to check that the sum of two outputs (where you have the normal output of a transaction and the "change" leftover amount returned to the sender) is negative. So if you put two large but positive values in the transaction the overflow is then only checked that it is less than or equal to the inputs.

Normally the inputs are equal to the outputs of a transaction. The exception is when there is a "fee" charged for the transaction. The net allows anyone to voluntarily pay any amout for a fee. SO when the sum was negative the difference from the input looked like a fee. It slipped thru all the checks. Her is some of the details:
 out Value 1:92233720368.54(7fffffffff85ee0)
 out Value 2:92233720368.54(7fffffffff85ee0)

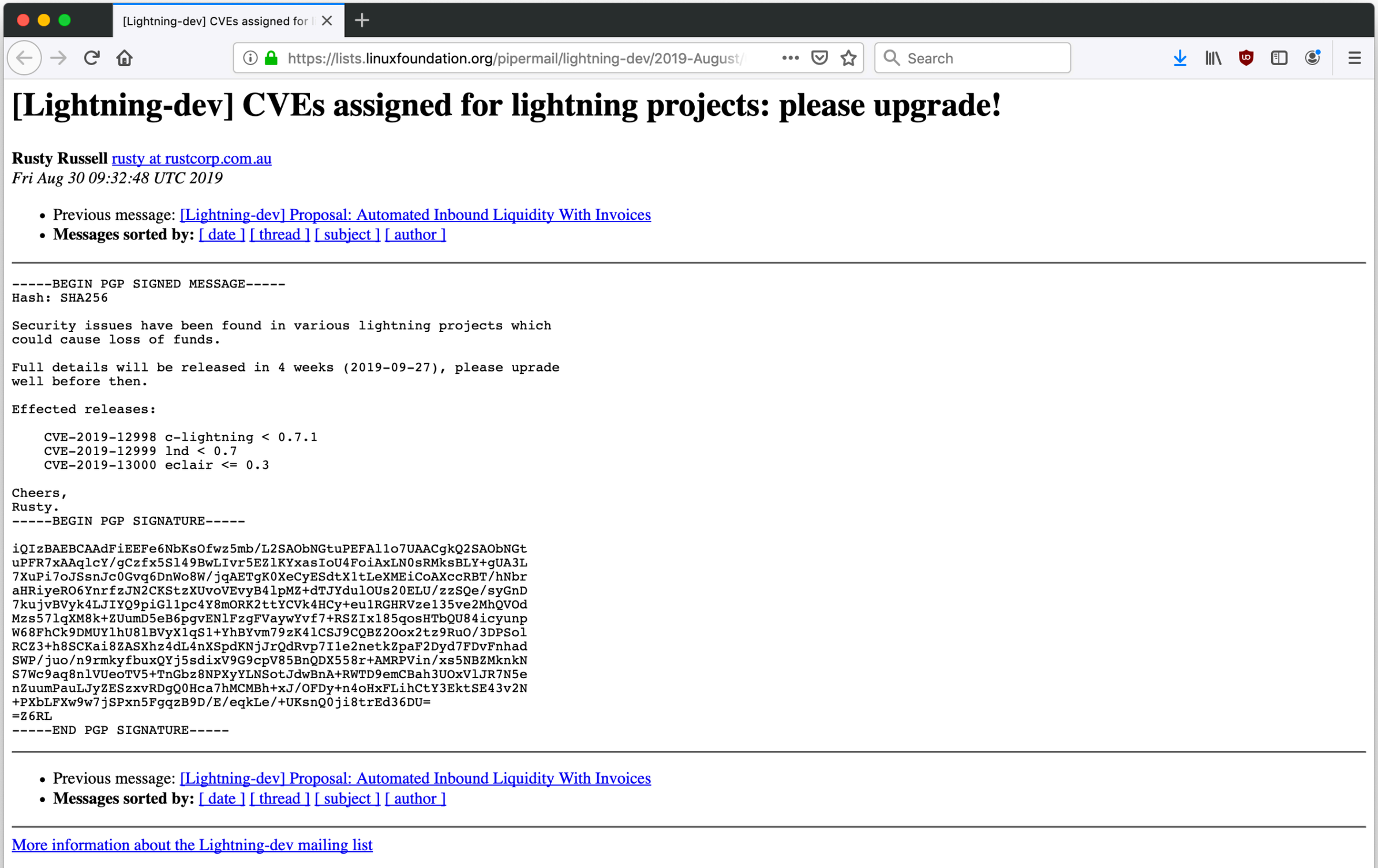
the sum would make -0.01 BTC

generated transaction "reward" including 51 bitcent "fee"
 out Value:50.51(000000012d1024c0)

that implies the input value was 0.50 BTC

We still have vulnerabilities today

Lightning



[Lightning-dev] CVEs assigned for lightning projects: please upgrade!

Rusty Russell rusty at rustcorp.com.au
Fri Aug 30 09:32:48 UTC 2019

- Previous message: [Lightning-dev] Proposal: Automated Inbound Liquidity With Invoices
- **Messages sorted by:** [date] [thread] [subject] [author]

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA256

Security issues have been found in various lightning projects which
could cause loss of funds.

Full details will be released in 4 weeks (2019-09-27), please uprade
well before then.

Effected releases:

    CVE-2019-12998 c-lightning < 0.7.1
    CVE-2019-12999 lnd < 0.7
    CVE-2019-13000 eclair <= 0.3

Cheers,
Rusty.
-----BEGIN PGP SIGNATURE-----

iQIzBAEBCAAdFiEEFe6NbKsOfwz5mb/L2SAObNGtuPEFAl1o7UAACgkQ2SAObNGt
uPFR7xAAqlcY/gCzfx5Sl49BwLIvr5EZlKYxasIoU4FoiAxLN0sRMksBLY+gUA3L
7XuPi7oJSsnJc0Gvq6DnWo8W/jqAETgK0XeCyESdtX1tLeXMEiCoAXccRBT/hNbr
aHRiyeRO6YnrfzJN2CKStzXUvoVEvyB4lpMZ+dTJYdulOUs20ELU/zzSQe/syGnD
7kujvBVyk4LJIYQ9piGl1pc4Y8mORK2ttYCVk4HCy+eu1RGHRVze135ve2MhQVOd
Mzs57lqXM8k+ZUumD5eB6pgvENlFzgFVaywYvf7+RSZIx185qosHTbQU84icyunp
W68FhCk9DMUYlhU8lBVyX1qS1+YhBYvm79zK4lCSJ9CQBZ2Oox2tz9RuO/3DPSol
RCZ3+h8SCKai8ZASXhz4dL4nXSpdKNjJrQdRvp7I1e2netkZpaF2Dyd7FDvFnhad
SWP/juo/n9rmkyfbuxQYj5sdixV9G9cpV85BnQDX558r+AMRPVin/xs5NBZMknkN
S7Wc9aq8nlVUeoTV5+TnGbz8NPXyYLNSotJdwBnA+RWTD9emCBah3UOxVlJR7N5e
nZuumPauLJyZESzxvRDgQ0Hca7hMCMBh+xJ/OFDy+n4oHxFLihCtY3EktSE43v2N
+PXbLFXw9w7jSPxn5FgqzB9D/E/eqkLe/+UKsnQ0ji8trEd36DU=
=Z6RL
-----END PGP SIGNATURE-----
```

- Previous message: [Lightning-dev] Proposal: Automated Inbound Liquidity With Invoices
- **Messages sorted by:** [date] [thread] [subject] [author]

More information about the Lightning-dev mailing list

Lightning

Lightning Labs⚡ @lightning · Sep 10, 2019

Replying to @lightning

Please upgrade to the latest versions, as they are not at risk. These are:

• lnd 0.7.1
• c-lightning 0.7.1
• eclair 0.3.1

Lightning Labs⚡
@lightning

This is also a great time to remind folks that we have limits in place to mitigate widespread funds loss at this early stage. There will be bugs.

Don't put more money on Lightning than you're willing to lose!

70 9:21 AM - Sep 10, 2019

59 people are talking about this

ZCoin Bug (2017)

How we discovered the bug

Slightly before midnight on February 16th GMT+0, during a check to analyze how widely used Zerocoin transactions were in Zcoin, we discovered that the total Zerocoin spend amounts did not tally with the number of mint transactions and the spend transactions far exceeded the number of mint transactions.

Upon further investigation, the dev team discovered that at block 11002, the serial number for the spend transaction had been reused which meant that someone was exploiting a single proof to generate multiple spends. If the code was working correctly, the duplicate serial numbers should have been rejected.

What we did following discovery of the bug

In the following few hours, our developers had identified that the issue was caused by a “==” sign being used instead of a “=” sign as linked here.

Source: https://zcoin.io/zcoins-zerocoin-bug-explained-in-detail/

ZCoin Bug (2019)

We found the root cause of the irregular Zerocoin spends on the 19 April 2019. An emergency update 13.7.9 is now available to disable Zerocoin completely while we move to our Sigma implementation. We are in touch with a number of other Zerocoin projects and are working together to secure it.

We recommend any projects utilizing Zerocoin (regardless of which implementation you are using) to disable Zerocoin on sporks or at a consensus layer.

A disclosure with the root cause of the fix will be released once we are satisfied that there is no longer any threat.

Source: https://zcoin.io/update-on-zerocoin-spends/

ZCash Bug

- Originated in fundamental 2013 cryptography paper
- Discovered by an engineer working for the company that developed ZCash
- Allowed infinite creation of coins
- People think it wasn’t exploited, but we don’t know
- Kept quiet, fix developed over many months, pushed, and then announced

V&V

- Validation: Are our specifications correct? Are we making the right thing?

- Verification: Did we faithfully implement the specification?

Which of the previous examples were which?

Back to Lightning...

A Composable Security Treatment of the Lightning Network

Aggelos Kiayias[1,2] and Orfeas Stefanos Thyfronitis Litos[1]

[1] University of Edinburgh
[2] IOHK
akiayias@inf.ed.ac.uk, o.thyfronitis@ed.ac.uk

Paper released in past week:

Abstract. The high latency and low throughput of blockchain protocols constitute one of the fundamental barriers for their wider adoption. Overlay protocols, notably the *lightning network*, have been touted as the most viable direction for rectifying this in practice. In this work we present for the first time a full formalisation and security analysis of the lightning network in the (global) universal composition setting that takes into account a global ledger functionality for which previous work [Badertscher et al., Crypto'17] has demonstrated its realisability by the Bitcoin blockchain protocol. As a result, our treatment delineates exactly how the security guarantees of the protocol depend on the properties of the underlying ledger. Moreover, we provide a complete and modular description of the core of the lightning protocol that highlights precisely its dependency to underlying basic cryptographic primitives such as digital signatures, pseudorandom functions, identity-based signatures and a less common two-party primitive, which we term a combined digital signature, that were originally hidden within the lightning protocol's implementation.

“Our analysis is based on the formal specification, not an implementation. As a result, our work does not rule out bugs in the various implementations, only in the specification…

Ideally, formal verification of the code, which would prove that it matches the specification, would increase our trust to the system. But before that, a machine-readable version of the specification would be needed.”

-Orfeas Litos

How to Judge Specification?

- Security analysis
- Game theory
- Simulation
- …

Test Cases

- Given an implementation, traditional testing with test cases is good

- But how do you know you’re testing everything you need to test? How confident can you really be that the implementation conforms to the specification?

Formal Verification

- Proving the correctness of a system with respect to its formal specifications or properties, using formal methods of mathematics

- Used for hardware or software
  - More for hardware. Why?

- Need a mathematical model of system that can then construct proofs within; several options

It’s difficult

Not Secure — compcert.inria.fr

HOME PARTNERS MOTIVATIONS RESEARCH COMPILER PUBLICATIONS DOWNLOADS

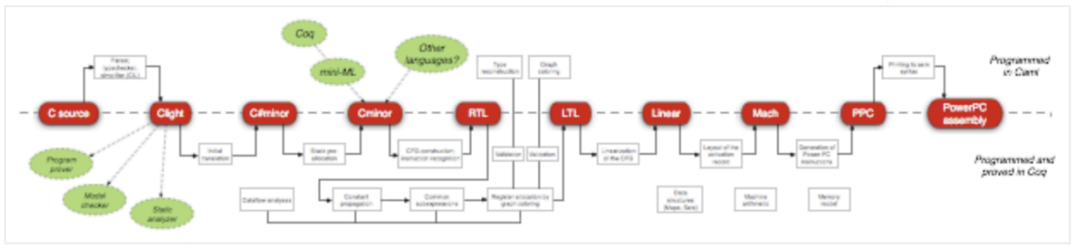
COMPCERT

COMPILERS YOU CAN *FORMALLY* TRUST

The CompCert project investigates the formal verification of realistic compilers usable for critical embedded software. Such verified compilers come with a mathematical, machine-checked proof that the generated executable code behaves exactly as prescribed by the semantics of the source program. By ruling out the possibility of compiler-introduced bugs, verified compilers strengthen the guarantees that can be obtained by applying formal methods to source programs.

The main result of the project is the CompCert C verified compiler, a high-assurance compiler for almost all of the C language (ISO C99), generating efficient code for the PowerPC, ARM, RISC-V and x86 processors.
Get CompCert C »

MENU



NEWS

[09/2019] CompCert C version 3.6 is released. Novelties include support for the AArch64 (ARMv8 in 64-bit mode) target architecture, and better support for generating branchless code, including a new if-conversion pass that turns conditional statements and expressions

Recall: For any Turing complete language, finding all possible runtime errors in an arbitrary program is undecidable

Does this make us think differently about Turing complete smart contract languages?

TLA+

- Created by Leslie Lamport
- A formal specification language for modeling programs and systems
- Especially suited for modeling concurrent and distributed systems
- Used by Amazon for AWS

```
People == {"alice", "bob"}
Items == {"ore", "sheep", "brick"}
(* --algorithm trade
variable owner_of \in [Items -> People]

process giveitem \in 1..3 \* up to three possible trades made
variables item \in Items,
          owner = owner_of[item],
          to \in People,
          origin_of_trade \in People
begin Give:
    if origin_of_trade = owner then
        owner_of[item] := to;
    end if;
end process;
end algorithm; *)
```

Source: https://learntla.com/introduction/

Deductive Verification

- Interactive proof assistants
  - HOL, Coq, Isabelle, etc.
  - Can often export to another language
- SMT (Satisfiability modulo theories) solvers
  - Constraint satisfaction
  - See Z3

Dependent Types

- What if a type's definition is dependent on a value?
- Example: A type not just for integers, but for integers less than 3
- What does this allow you to do at compile time?
- Languages: Agda, Coq, F*, Idris, and more
  - It’s not surprising if you haven’t heard of any of these
- Curry-Howard Correspondence

Dependent Types

- What if a type's definition is dependent on a value?
- Example: A type not just for integers, but for integers less than 3
- What does this allow you to do at compile time?
- Languages: Agda, Coq, F*, Idris, and more
  - It’s not surprising if you haven’t heard of any of these
- Curry-Howard Correspondence

Other Worthwhile Mentions

- Penetration testing
- Audits
- Many eyes

Final Projects

- Poster session and brief report
- Work alone or group of up to 3
  - All members of a group get the same grade
- Choose something you find interesting
  - But ask for help if you struggle getting an idea
- You have many options
  - Implement a system, for example an interesting smart contract, a protocol, a game, a key management system, etc.
  - Conduct research, for example design an algorithm, design a protocol, benchmark existing systems, perform cryptographic analysis, write a specification, formally verify some open source code, etc.
  - Survey some area of technology
- Check your project ahead of time in office hours to verify appropriateness of scope