AUDIT REPORT:

Audit of the CloakCoin PoSA3 implementation

Iain Craig <coldcity@gmail.com>, 28 July 2015

I. BACKGROUND

A majority of cryptocurrency implementations are open source, with the full codebase publically available. It has been well documented that this allows for a highly effective peer review process; to quote Eric S. Raymond, "given enough eyeballs, all bugs are shallow"[1].

In the case of the CloakCoin codebase, it is not desired to fully disclose the source code at this time, in order to prevent a commercial advantage being gained by competing "clone" coins adopting the CloakCoin anonymization technology, PoSA3.

It is still, however, extremely important to the CloakCoin team to be as transparent as possible, within this limitation. For this reason, a white paper[2] has been released by the CloakCoin team explaining in detail the mechanics of the PoSA3 algorithm and associated transaction processes. The PoSA3 algorithm itself is thus public knowledge, and peer review of this algorithm continues in the public eye.

It was determined that in addition to this, an internal audit of the codebase should be performed, and that this process should be conducted in the open, although the codebase itself cannot yet be opened.

The audit has been conducted by Iain Craig, a consultant to the CloakCoin development team who has not been actively involved in the development of PoSA3.

Iain is known in the cryptocurrency community as Jonny Bravo. He is a degree-educated professional developer with over 17 years of commercial experience in all aspects of software development, as well as extensive expertise in network management and penetration testing. He has been involved in the development of several altcoins and numerous products and services in the cryptocurrency ecosystem and offers general cryptocurrency consultancy services.

Iain is trusted by the CloakCoin team to fully examine the CloakCoin source code without limitation, and was provided complete access to the CloakCoin team and developers while conducting this audit. Not being directly involved in the development of the PoSA3 code has allowed him to bring fresh eyes to the audit process.

---

[1] http://www.catb.org/esr/writings/cathedral-bazaar/
[2] http://cloakcoin.com/downloads/posa3wp.pdf

II. AIMS

It was the goal of this audit to examine and validate the current codebase in four core ways.

(1) **Algorithmic accuracy**: How closely the codebase adheres to its stated intentions, i.e. how correctly and efficiently it implements the PoSA3 algorithm as presented in the white paper.

(2) **Algorithmic quality**: How effective the PoSA3 algorithm is in achieving its objective of anonymising transactions.

(3) **Codebase quality**: How well the implementation has been performed in terms of efficiency, adhesion to general programming best practices, and consistency of style. This latter item encompasses both consistency internal to the new work, and conformity of the new work to its wider environment; i.e., how well it follows the pre-existing practices already present in the pre-existing code. These practices may be by design, or idiomatic.

(4) **Codebase security**: How secure the codebase is. This is examined with particular attention to any potential backdoors in the wallet, whether accidentally or deliberately placed, and to any potential new attack surface added to the wallet through vulnerabilities introduced by the new work (again whether accidental or deliberate).

III. LIMITATIONS

It must be stressed that this is an internal audit. Therefore the conclusions drawn, while made by a consultant to the team, are ultimately those of the team itself; therefore this audit does not remove the requirement for end users to trust the CloakCoin team.

Note that a thorough examination of the security and effectiveness of the PoSA3 algorithm itself is out of the scope of this audit. This protocol is publically available and described in the white paper; an ongoing process of peer review continues to examine the algorithm and offer refinements and improvements, and indeed the current state of play of the white paper has been revised iteratively during this process. Therefore, Aim 2 above is addressed mostly by the peer review process rather than this audit.

Note also that Mac binaries were not built and tested as part of this audit.

Finally, please note that this audit concentrated primarily on the new code introduced by the PoSA3 work, rather than the (publically available) codebase prior to the current CloakCoin team taking over maintenance of the code.

IV. METHODOLOGY

Each aim above required the use of different toolsets and analysis techniques, albeit with much overlap between the aims.

A fundamental technique employed was manual source code analysis. As it was the intention to focus specifically on all new work performed since the current CloakCoin team took over maintenance of the codebase, a series of milestone snapshots of the codebase were decided upon, with a principle methodology being to compare the source code differences between these snapshots. The snapshots used were as follows:

Snapshot A: The last public source release of CloakCoin before the current team took over maintenance. This remains available at https://github.com/CloakCoin/CloakCoinRelaunch/commit/9f318b895ebd7babc9324475d539d79c33764bc4.

Snapshot B: The WIP PoSA1 development branch as handed over by previous maintainer Alty. Note: This was presented in a highly incomplete state and was never publically released by Alty.

Snapshot C: The WIP PoSA3 development branch as handed over by Alty. This too was presented in an unfinished state and was never publically released.

Snapshot D: CloakCoin PoSA3 Beta 1 (Internal codename cpa3-1.9.7.46)

Snapshot E: CloakCoin PoSA3 RC1 (Internal codename cpa3-RC1)

As well as tracking the source code differences from snapshot to snapshot, the evolution of the PoSA3 algorithm itself, as documented in various internal and public versions of the whitepaper, was tracked in parallel to check for divergence of one from the other.

In addition to extensive manual source code review, binary analysis of Snapshot D and Snapshot E was performed.

This involved hosting compiled binaries on VMware virtual machines running Windows 7 and Linux (vanilla Debian Squeeze). The binaries were hence placed in a sandbox environment where all filesystem and network activity could be monitored, as well as all system calls invoked by the running wallets. Tools used to monitor the behaviour of the binaries included Wireshark[3], Sysmon[4], API Monitor[5] and Process Monitor[6] on Windows. Tools used for the same on Linux were strace[7], tcpdump[8], iptraf[9] and FAM[10].

[3] https://www.wireshark.org/
[4] https://technet.microsoft.com/en-gb/sysinternals/dn798348
[5] http://www.rohitab.com/apimonitor
[6] https://technet.microsoft.com/en-gb/sysinternals/bb896645
[7] http://sourceforge.net/projects/strace/

For some tests, the binaries were allowed to connect to a closed testnet comprised of 4 identical clients on virtual machines, able to connect to each other but with no connection to the outside world.

As well as behavioural monitoring of the binaries, other binary analysis performed included the use of the Valgrind[11] framework to check memory management and threading issues, with particular attention to memory leaks.

Input fuzzing was used in attempts to break the binary wallet, subvert its state or expose any vulnerable attack surface. Particular attention was paid to attempting to overrun buffers, and inject malformed packets.

In analysing the PoSA3 algorithm itself, state diagrams and entity modelling were used. In order to ascertain how closely the codebase implements PoSA3, cflow[12] and gprof[13] were used to generate call graphs and examine functional dependencies.

[8] http://www.tcpdump.org/
[9] http://iptraf.seul.org/
[10] http://oss.sgi.com/projects/fam/faq.html
[11] http://valgrind.org
[12] http://www.gnu.org/software/cflow/
[13] http://www.gnu.org/software/binutils/

V. FINDINGS & RECOMMENDATIONS

1. Algorithmic accuracy

Taking the white paper as a specification, the algorithm is implemented exactly as described. In white paper sections 2.2, the algorithm is presented as a set of discrete steps. Each step presented can be traced to a reasonably discrete block of code in the codebase. There are no unexplained extra steps taken in the code; the algorithm described in the white paper is fully implemented as given.

No issues were found in this area, and no recommendations are made.

2. Algorithmic quality

As described, the fact that the white paper detailing the PoSA3 algorithm is publically available allows for proper peer review of the PoSA3 anonymization techniques. It is felt that this is the best way to prove the algorithm itself, completely in the open and subject to analysis by any interested party.

However it is the opinion of the auditor that the PoSA3 algorithm as presented in the white paper is effective at concealing the sender and recipient of a coin transfer in a “trustless-anonymous” fashion, to use the terminology of the paper.
This is borne out by extensive process modelling and demonstrative testing, both in a closed system during the audit process and as part of the development lifecycle within the CloakCoin team. Both internal and invitee-only testing phases, and ultimately a public beta testing phase, have shown that it is impossible to pick out the senders and recipients of transactions by scrutinising the blockchain.

Furthermore, the issues with participant trust in the predecessor to PoSA3 (the PoSA1 algorithm) have in the opinion of the auditor been resolved. The combination of all inputs and outputs into a single transaction is an elegant solution to the threat of bad actor participants in the cloaking process.

That being said, there are a small number of areas which the auditor feels could be improved. These are laid out below. However, it must be stated that these are in no way issues that break the PoSA3 process; rather, they are recommendations for refinements and improvements that improve the workflow and overall success rate, and provide for yet further anonymization of transactions.

ISSUE 1: Unencrypted initial announcement causing potential leak of Session Public Key

Explanation: PoSA3 nodes joining the network currently send their initial PoSA3 announcement (containing the Session Public Key) using the standard CloakCoin messaging

channel, rather than Onion Routing over the secure encrypted CloakShield channel. This is due to CloakShield and Onion Routing not being available at the moment when a new node joins the PoSA3 network; the node is not yet connected to other PoSA3 peers and therefore cannot participate initially in secure communications. The need to securely route the initial PoSA3 announcement using CloakShield and CloakShield's requirement of already being connected to PoSA3 peers are currently mutually exclusive.

Recommendation: PoSA3 broadcasts could initially be flagged as 'not available for cloaking' so they can negotiate securely with nodes currently on the PoSA3 network to initially connect, as this process directly leaks their Session Public Key due to onion routing not being available at initial connection (not enough PoSA3 nodes). To mitigate this, the initial broadcast could be discarded as soon as the node has connected to enough PoSA3 nodes to securely onion route. The initial PoSA3 announcement can then be left to expire and be superseded by a new announcement (now flagged as 'available for cloaking') which is onion-routed out across the PoSA3 network. This would allow the node to facilitate onion routing with the initial announcement without leaking details by not participating in PoSA3 Cloaking operations with those initial credentials.

ISSUE 2: Lack of CloakShield receipt messages

Explanation: Nodes that receive, decrypt and decode a CloakShield packet successfully do not send a receipt back to the sender. This reduces the scope for successfully detecting failed onion routing attempts (due to PoSA3 routing nodes dropping offline) and does not allow the message to be resent using an alternate route.

Recommendation: Peers should send a response to indicate that they have successfully received a CloakShield message. This can then be used to circumvent any onion routing issues and allow the sender to automatically resend the message using a different route when necessary.

ISSUE 3: Potential for additional PoSA status messages

Explanation: During the PoSA3 negotiation process, both the PoSA3 Sender and participating Cloakers perform a series of checks on the inputs and outputs that comprise a PoSA3 transaction. If any of the participating parties detects an anomaly, they immediately cease their participation in the PoSA3 transaction, effectively aborting and voiding the PoSA3 transaction, which leaves the Sender needing to manually retry the send.

Recommendation: Additional PoSA3 negotiation messages could be added to indicate that:

a. A participating node sent bad input/output data and Sender needs to dismiss the Cloaker and use a replacement peer. In this instance, the Sender would re-broadcast the request advertising availability for new participants or use a previously cached PoSA3 acceptance from a potential Cloaker.

b. A Cloaker received a PoSA3 transaction for signing which failed the validity checks. In this instance, the Cloaker would inform the Sender of the issue and

Sender can automatically amend or recreate the PoSA3 transaction for signing. In this scenario the Cloaker could penalize the Sender with a DoS penalty.

c. The finalized PoSA3 transaction failed validation after signing due to an attempted 'double spend' which caused the network to reject the PoSA3 transaction. The existing validation code in the PoSA3 framework will allow the Sender to identify which Cloaker supplied the bad inputs/outputs. In this instance, the Sender would re-broadcast the request advertising availability for new participants and could penalize the Cloaker at fault with a DoS penalty.

ISSUE 4: No re-negotiation of failed PoSA3 transfer

Explanation: If a PoSA3 transfer fails due to a participant aborting or failing to respond, there is currently no methodology in place to automatically re-negotiate the creation of a PoSA3 transaction. Adding this functionality would circumvent the need for manual re-sending of failed PoSA3 transactions and provide a much smoother user experience for the PoSA3 process.

Recommendation: This recommendation is covered by the additional messages outlined in *ISSUE 3: Potential for additional PoSA status messages* above.

3. Codebase quality

The PoSA3 implementation has been done extremely well. The new PoSA3 code perfectly follows the style of the pre-existing code, from aspects such as brace style and variable naming conventions through to the higher level philosophy. The level of commenting is good, and classes and members are logically named.

In particular the new code does a good job of not "reinventing the wheel"; where goals can be accomplished using existing features of the codebase, these features have been utilised well.

A good example of this is the networking subsystems to facilitate PoSA3 transactions; particularly in the case of the CloakShield subsystem (see white paper). Additional code has been required to implement the ECDH key exchange and RSA stream cypher system. This is done in a very sympathetic way, using existing functions to perform these tasks where possible and importing a minimum of library code.

Only two issues were identified in this area, as follows.

ISSUE 5: Missing public key validation

Explanation: The generation of the EDCH shared secret involves the public key submitted by a PoSA3 node. This public key is not validated, and a possible attack vector is exposed. The client can currently be crashed by malformed public keys; it's unclear whether there is a potential for code injection but this is a possibility. In any case, a DDoS attack against the

network would be possible by a malicious client deliberating sending malformed public keys and forcing recipient nodes offline.

Recommendation: Check the public keys received from PoSA3 nodes are valid before generating the ECDH shared secret.

ISSUE 6: Incorrect locking of 'Stake' and 'PoSA Processing' funds

Explanation: It has been noted that the system in place for reserving coins in the wallet for staking and PoSA3 participation will occasionally reserve too many coins for staking.

Recommendation: This is a trivial code fix in the coin reservation subsystem.

4. Codebase security

This really encompasses two facets. Firstly, is the code implemented in such a way as to not create any additional attack surface area on the application? Secondly, can the developer be trusted to not deliberately create any backdoors?

The codebase was analysed with both these aspects in mind. It was checked that all allocated buffers are of appropriate size and, crucially, include bounds checking. Furthermore, fuzzing testing ascertained that the code robustly handles malformed packets and other malformed testing. The only exception to this is noted in section 2 above, under "Missing Public Key Validation".

Although Boost smart pointers or a similar system can be used within a codebase in order to enforce correct object dereferencing, this technique is not used within the PoSA3 or wider CloakCoin code. This is often suggested as a best practice, but in the view of the auditor is not required in this case. Memory management is robustly handled by the base code and the new PoSA3 code; Valgrind was used to determine that there are no memory leaks, and retrofitting such a deep change would require touching a vast amount of base code which is already known to work correctly. For this reason, introducing smart pointers is not recommended in the case of the CloakCoin codebase.

Through manual code analysis, as well as behavioural analysis of the compiled Windows and Linux binaries, it can be stated with a very high degree of confidence that there is no malicious code in the codebase. Manual analysis alone is not considered by the auditor to be sufficient to make such a claim; obfuscated code can be inserted into a codebase by a skilled programmer and split up in such a way as to appear completely innocuous. In the case of a complex application, and particularly an application which depends for a large part on network –related code, malicious code can hide from view even when one is able to survey a fully expanded call graph.

For this reason, the binaries were monitored in order to observe all system calls, all network packets and all filesystem access. The tools used are outlined in the methodology presented above. Please note that this binary analysis was conducted having first determined that the

codebase does not contain any code to check whether the application is executing inside a virtual machine.

It was found that no files are touched apart from the CloakCoin wallet and associated config and cache files; no spurious network packets are generated; no back doors are opened on the local machine; no inexplicable listening ports are opened; no suspicious system calls are seen.

Absolutely nothing untoward, nor out of keeping with the mechanics of PoSA3 transactions, was observed from this binary analysis.

No recommendations resulted from examination of codebase security, although it could be argued that the recommendation of public key validation given above partially falls under this category.