

Security Audit Report

Classification: Confidential
Document last revised on: 2018-01-19



TABLE OF CONTENTS

1	EXECUTIVE SUMMARY	3
1.1	Findings	3
1.2	Recommendations	3
2	INTRODUCTION	4
2.1	Background	4
2.2	About Cognosec GmbH	4
2.3	Objective	4
2.4	Scope	4
2.5	Out of Scope	5
2.6	Testing Narratives	5
2.7	Disclaimer	5
3	METHODOLOGY	6
3.1	Identification and Analysis	6
3.2	Vulnerability Detection	6
3.3	Evaluation	7
3.4	Reporting	7
3.5	Standards	7
3.6	Quality Assurance	8
3.7	Code of Ethics	8
4	CONTACT LIST	9
5	ACTIVITIES AND DELIVERABLES	9
6	SEVERITY LEVELS	10
7	FINDINGS	11
7.1	Vulnerabilities	11
8	ABBREVIATIONS	26
9	APPENDIX	27
9.1	Evidence	27

1 EXECUTIVE SUMMARY

This report reflects the results of the security audit of Cloak as of January 2018. The evaluation was conducted to identify vulnerabilities and weaknesses that could be misused by attackers. The scope was defined as the Cloak cryptocurrency wallet application, the underlying blockchain mechanisms and system as well as especially the Enigma technology that aims to provide an additional level of anonymity.

The majority of security controls were tested manually following a standardised approach; repeated tasks were automated where possible. Identified security issues were reviewed to eliminate false positives, prioritised according to related risk, and measures for their remediation were proposed.

The results of the assessed areas led to the impression that the basic mechanisms are quite robust from an application security perspective, but the implementation of Enigma should be reviewed and improved. Some instances of implementation weaknesses were identified which in some cases are contradicting the concept as described in the official Enigma whitepaper.

The following subchapters list the most important findings identified and suggested remediation tasks. Additional information and findings with lower associated risk are provided in the subsequent chapters of the report.

1.1 Findings

- Due to implementation flaws it was possible to conclude on the amount of transferred cloaks and subsequently on the input and output addresses of sender and cloakers of transactions using a low number of cloakers.
- Even though wallet encryption is enabled by a user, the transactions are stored unencrypted and, as a consequence, might get extracted, which has an impact on the anonymity of the owner.
- An old version of the Bitcoin and the Tor code was identified as underlying base version of the Cloak source code. Therefore, Cloak inherits some of the vulnerabilities Bitcoin and Tor had since this version.
- Some functions, like a random number generator in use, did not perform as intended and therefore decreased the level of anonymity of Enigma transactions.
- The static source code analysis showed multiple instances of methods that are deprecated and banned as they are unsafe and might have an impact on the security of the application.

1.2 Recommendations

- It should not be possible to determine the amount of transferred cloaks and the reward should not be split equally to prevent from distinguishing between cloakers and sender.
- When encryption is enabled, wallets should encrypt all transaction details stored, including the transaction history.
- The potentially vulnerable functions are deprecated and should not be used any longer.
- The Cloak application should be adapted to use the latest version of the Bitcoin code base.
- The affected functions should be adapted according to the detailed recommendations in the finding section.

2 INTRODUCTION

Cognosec has conducted a security audit of the Enigma technology and the Cloak wallet application, as defined in "Scope". The results of the assessment are covered in this document. Actual security testing started on the 18th of December 2017 and was concluded on the 12th of January 2018. The objective of the assessment was to pinpoint security weaknesses and vulnerabilities, and to propose recommendations for their remediation.

Issues were discovered using targeted manual security testing procedures that were backed up with tools that allow automation of certain tasks. The identified issues were evaluated and prioritised according to their relative risk and measures for their remediation were proposed. The proposed countermeasures to reduce risk for identified security issues are presented in this security assessment report.

2.1 Background

Cloak went open source end of December 2017. In addition, the introduction of an improved version of the Enigma technology is planned that aims to provide an additional layer of anonymity. As a consequence, Cloak decided to conduct an external security audit in order to obtain assurance that the application is mature from an application security perspective.

2.2 About Cognosec GmbH

Cognosec GmbH is headquartered in Vienna, Austria and is a member of the Cognosec AB (Publ) group of companies. Cognosec GmbH offers services in information security, governance, enterprise risk management, compliance, and assurance to clients. Our solutions are based on domain knowledge in finance, telecommunications, online gaming and e-commerce industries in Europe, Africa, the United States and the Middle East. Cognosec adds value through the deployment of professional and management consultancy services that fit the corporate risk appetite and budget of its clients.

2.3 Objective

The aim of the assessment is to provide an independent and reliable opinion on the security of the Cloak application and specifically on the Enigma technology. The assessment shall identify weaknesses and vulnerabilities and quantify their severity so they can be managed and addressed and therefore help

- Preventing from malfunction and/or financial loss through fraud or unreliable infrastructure;
- Providing due diligence to regulators, customers and shareholders and
- Protecting the brand against reputation loss.

2.4 Scope

A security audit test was performed utilising the white-box test which included:

- Source Code Review
- Enigma Process Analysis

Following a risk-based approach, the audit focused mainly on the following components of the Cloak application:

- Cloak Wallet
- Enigma

2.5 Out of Scope

Due care was taken in order not to damage Cloak property, not to have an impact on systems or to interfere with Cloak's daily business. Specifically the following approaches were not in scope of the assessment, however, situations that would allow such proceedings would have been documented:

- Denial of Service attacks
- Tampering with information integrity
- Mathematical analysis of involved cryptographic algorithms and methods

2.6 Testing Narratives

The audit team has been provided with the source code of the Cloak application in version Cloak2-2.1.0. In addition, compiled versions of this application for the use in testnet3 and testnet5, as well as access to the corresponding blockchain explorer has been provided.

Using the approach defined in the Methodology section, the source code of the Cloak Wallet, as well as the processes involving cloaked transactions via the Enigma technology, were tested.

2.7 Disclaimer

All the assessment work undertaken for this report has been provided by certified professionals in accordance with good industry practise and in line with all obligations and regulations imposed by the various relevant certification bodies.

The information in this report is subject to and limited by the conditions as described in the scope and objectives sections being such agreed upon conditions and objectives to determine the scope of the activities undertaken to derive this report.

In any authorised audit or assessment, time and resources are naturally limited and so when compared to the potentially unlimited time and resources available to parties with malicious intent, the existence of vulnerabilities and weaknesses will be verified but the non-existence of any and all vulnerabilities cannot be assured absolutely.

In this context, while every effort has been made to audit and assess the system using our best skill, knowledge and belief, this report in no way guarantees the establishment of an impenetrable system. As a result, neither Cognosec nor any of its group companies, employees or consultants will be liable for any direct or indirect loss or damage caused by any failure or breach of an organisation's information technology systems in which the information in this report was used or relied upon.

The information in this report is intended for use by the recipient company only and neither Cognosec nor any of its group companies, employees or consultants will be liable for any direct or indirect loss or damage caused by any other person's or entities' reliance upon on such information.

3 METHODOLOGY

The following steps are conducted to deliver an independent and professional opinion in regards to effectiveness and adequacy of the security controls of the information systems:

- Threat Identification: Identification of threats and potential attack surface
- Vulnerability Detection: Evaluation of current security posture
- Evaluation: Evaluation and prioritization of the identified weaknesses and vulnerabilities
- Exploitation: Exploitation of identified vulnerabilities to demonstrate potential impact to confidentiality and integrity
- Reporting: Determination and reporting of appropriate measures to eliminate or minimize risk



Figure 3.1: Testing Phases

3.1 Identification and Analysis

The first phase of the assessment focuses on gathering, analysis and structuring of information about the items in scope, mainly utilizing passive analysis techniques. Also public sources such as websites, blogs and search engines are queried to retrieve valuable information about the target environment. This is done to identify the attack surface of the environment and gather needed information to conduct the following testing and exploitation phases. Potential threats are identified and ranked according to their risk to adopt and direct the manual testing procedures.

3.2 Vulnerability Detection

Automated and manual testing approaches are combined to cover the majority of potential vulnerabilities. Utilizing static and dynamic code analysis increases the detection range of identified common known security vulnerabilities. By manually testing critical aspects, utilizing a predefined methodology, security flaws that are not covered by the automated testing approach can be uncovered. Besides evaluation against general industrially accepted best practices, the following critical functions will be inspected in detail:

- Behaviour of the application (communication channels, used protocols)
- Architectural Review

- Secure generation of keys / seeds used in the cryptocurrency system
- Secure wallet creation
- Secure storage of cryptographic keys
- Secure usage of cryptographic keys
- Processes and procedures related to key compromise protocol (KCP)
- Keyholder Grant/Revoke Policies & Procedures
- Applicable auditing/logging capabilities

3.2.1 Static Analysis

Static source code analysis is used to cover the entire code base and identify all the vulnerable patterns. In static code analysis the entire code base is abstracted and all code properties and code flows are exposed. The result of this analysis is reviewed by application security experts to suppress false positives and reprioritize identified issues based on the severity and imposed risk of the issue.

3.2.2 Dynamic Analysis

Dynamic analysis involves examining the app from the outside while executing it. This type of analysis can be performed manually or automatically. It usually does not provide the information that static analysis provides, but it is a good way to detect interesting elements (assets, features, entry points, etc.) from a user's point of view. The focus of dynamic analysis is the testing and evaluation of apps via their real-time execution. The main objective of dynamic analysis is finding security vulnerabilities in a program while it is running. Dynamic analysis is usually used to check for security mechanisms that provide sufficient protection against the most prevalent types of attack, such as disclosure of data in transit, authentication and authorization issues.

3.3 Evaluation

Results from manual and automated analysis are verified for completeness and reasonability to decrease the risk of unidentified vulnerabilities, also called false-negatives, to an acceptable level. Findings are evaluated and reassessed each by each to verify they in fact represent vulnerabilities. The Common Vulnerability Scoring System Version 2 (CVSS v2) base score is assigned to the findings to categorize their impact and exploitability. That scoring system has been described by the National Institute of Standards and Technology (NIST) and is consistently adopted by Cognosec GmbH regarding vulnerability reporting.

3.4 Reporting

The client is regularly informed about status and progress of the assessment work. This regular status update consists of a summary of the overall progress and information about any issues interfering with the achievement of the assessment objective. In the case of imminent danger, the client is informed without delay as to prevent damage.

The results of the assessment are documented and delivered in the form of an assessment report. The assessment report contains an executive summary, outlining overall risk posture of the environment as well as key findings, a summary of the environment in scope, a description of the assessment methodology and the assessment work conducted and a detailed list of findings and recommendations.

3.5 Standards

Assurance work will be conducted in accordance to the "Information Technology Assurance Framework" (ITAF), a recognized standard for conducting IT assurance, issued by the "Information Systems Audit and Control Association" (ISACA). Industry security standards for information systems that make use of cryptocurrencies are followed from a technical perspective, specifically the "Cryptocurrency Security Standard" (CCSS).

3.6 Quality Assurance

Planning fieldwork, as well as reporting, is led by experienced and certified experts only. The comprehensive quality assurance process is executed in parallel to the assurance phases and the assigned quality manager checks the results of every single phase for completeness and accuracy before advancing to the next phase.

3.7 Code of Ethics

Cognosec auditors apply and uphold the following principles:

- **Integrity:** The integrity of auditors establishes trust and thus provides the basis for reliance on their judgment.
- **Objectivity:** Auditors exhibit the highest level of professional objectivity in gathering, evaluating, and communicating information about the activity or process being examined. Auditors make a balanced assessment of all the relevant circumstances and are not unduly influenced by their own interests or by others in forming judgments.
- **Confidentiality:** Auditors respect the value and ownership of information they receive and do not disclose information without appropriate authority unless there is a legal or professional obligation to do so.
- **Competency:** Auditors apply the knowledge, skills, and experience needed in the performance of audit services.

4 Assessment Team

Title	Role	Qualifications
Principal Auditor	Audit Supervisor, QA	CISA, GWAPT, QSA, ASV
Team Leader - IS Audit	Auditor	Dipl.-Ing., CISA, ASV
Senior IS Auditor	Auditor	Dipl.-Ing., CISA, ASV
IS Auditor	Auditor	B.Comp (Hons)

5 ACTIVITIES AND DELIVERABLES

Activity and Deliverable	End / Delivery Date
Assessment Start	2017-12-18
Assessment Complete	2018-01-12
Reporting Complete	2018-01-17
Quality Assurance	2018-01-18

6 SEVERITY LEVELS

The severity levels used in the Findings section to categorize the impact and exploitability of vulnerabilities adhere to the Common Vulnerability Scoring System Version 2 (CVSS v2) by the National Institute of Standards and Technology (NIST) . Reports use the base score that is composed by the type of access, the access complexity and required level of authentication to exploit a vulnerability as well as related impact on confidentiality, integrity and availability. The score applied to vulnerabilities ranges from 0 to 10 and is normalized by categorizing them into critical, high, medium and low severity levels. In addition, the exact vector is provided that is used to calculate the specific score in order to ensure transparency. Summarized, the vector is built upon the following metrics:

Access Vector (AV) : The access vector describes the required source of attack in order to exploit a vulnerability. Possible values are Local (L), Adjacent Network (A) or Network (N)

Access Complexity (AC) : The number or complexity of conditions that need to be in place for successful exploitation. Possible values are High (H), Medium (M) and Low (L).

Authentication (AU) : The number of authentication levels an attacker needs to pass in order to exploit a vulnerability. Possible values are Requires Multiple Instances (M), Requires Single Instance (S) and None Required (N).

Confidentiality (C), Integrity (I), Availability (A) : The impact on CIA is described. Possible values are None (N), Partial (P) and Complete (C).

Severity	Description
Critical	<ul style="list-style-type: none"> CVSS v2 Base Score 10 Exploitation is trivial Complete loss of a systems confidentiality, integrity and availability Immediate remediation is business critical
High	<ul style="list-style-type: none"> CVSS v2 Base Score 7 - 9.9 Exploitation nearly trivial Complete loss of at least one of C, I or A Remediation is business critical
Medium	<ul style="list-style-type: none"> CVSS v2 Base Score 4 - 6.9 Exploitation possible and common, requires skills Serious impact on CIA Corrective actions required within reasonable timeframe
Low	<ul style="list-style-type: none"> CVSS v2 Base Score 0.1 - 3.9 Exploitation possible but difficult and unlikely Measurable impact on CIA Corrective actions are nice to have
Informational	No actual vulnerability has been identified, but there is some information that might be of interest.

7 FINDINGS

7.1 Vulnerabilities

Vulnerability ID	1
Severity	High
Title	Compromise of Anonymity
CVSS Vector	AV:N/AC:L/AU:N/C:C/I:P/A:N
CVSS Score	8.5
Assets	<ul style="list-style-type: none">• Cloak Application
Description	<p>It was possible to reverse the anonymity provided by Enigma at a transaction that used three cloakers and as a consequence the sender, the recipient and the transferred amount could be determined.</p> <p>The same approach worked partially on transactions that used more cloakers, it was still possible to determine the transferred amount of cloaks at a transaction that used seven (six) cloakers. Due to the number of cloakers, an attempt to identify the senders output addresses led to six possible variations.</p> <p>The issue can be tracked down to the following weaknesses:</p> <ul style="list-style-type: none">- Certain output amounts of cloakers match the transferred amount- The reward is equally split between the cloakers
Remediation	The transaction outputs should be built in a way that it is not possible to determine the transferred amount of cloaks.
Details	<p>An example of reversing the anonymity added by Enigma (Transaction id 7689984b6f218ccb67ddc403d0d87ea3ad355854ca1e4c477086ab3eba6c015d of testnet5) can be found in the appendix.</p> <p>Please refer to figure 9.1 on page 27 for evidence.</p> <p>Please refer to figure 9.2 on page 27 for evidence.</p>

PROBLEM #1

Compromise of Anonymity is the largest issue where the transaction amount and the sender could be determined. It has been resolved. By analyzing the transactions on the blockchain provided for by Cloak, Cognosec found it was possible to determine the transaction amount by summing transactions searching for equal amounts through a method called possibility analysis. Once the transaction amount was solved for, it could be determined what the sender addresses were.

Based off Cognosec suggestions, we have implemented the following solution. Instead of an equally split Enigma fee, Cloaker participants receive incentives from Enigma fees which are randomly split, 80%-120%, of an equally split Enigma fee. Additionally, the transaction amount is repackaged and is then re-split 2-4 times in a way to prevent any equivalent transaction summations. This prevents using possibility analysis to determine the amount sent. Without the ability to determine the transaction amount, it is not possible to determine who sent the coins. Additionally, included in this solution is resolved Problem #9 – Flawed Splitting Randomizer

Vulnerability ID	2
Severity	High
Title	Insufficient Wallet Encryption
CVSS Vector	AV:A/AC:L/Au:N/C:C/I:C/A:N
CVSS Score	7.8
Assets	<ul style="list-style-type: none">• Cloak Application
Description	<p>If an attacker is able to steal a wallet (wallet.dat), the attacker is able to mount this wallet into their application without the need of a password even if the wallet was encrypted.</p> <p>While the attacker is still unable to initiate transactions, he is able to view the transaction history and therefore compromises the anonymity of the owner.</p>
Remediation	The whole wallet, including all transaction data, should be encrypted so unauthorized access to sensitive transaction data is not possible without providing the correct password.
Details	Please refer to figure 9.3 on page 28 for evidence.

PROBLEM #2

Insufficient Wallet Encryption is currently under development. Cognosec also found that even should the wallet be stolen, the coins could not be used. The current risk is that the transaction history would be known.

Vulnerability ID	3
Severity	Medium
Title	Random generator used without seed
CVSS Vector	AV:N/AC:L/AU:N/C:P/I:P/A:N
CVSS Score	6.4
Assets	<ul style="list-style-type: none">• Cloak Application
Description	<p>The <code>random_shuffle</code> function is called 8 times in the entire project but it is called only once with a random number generator supplied (line 1485 in <code>wallet.cpp</code>). Since the other 7 times are called without a random number generator supplied, <code>srand</code> will be used to seed the function. <code>srand</code> is seeded once in the project, found in the function <code>CWallet::GetCloakingOutputs</code> in <code>wallet.cpp</code> (line 3915). However, the function <code>CWallet::GetCloakingOutputs</code> is commented out and not in use, leaving the <code>random_shuffle</code> function unseeded in the 7 times it is used.</p> <p>If <code>random_shuffle</code> is not seeded, inputs and outputs of enigma transactions will not be truly randomized. Additionally, since the function is also used to shuffle nodes to relay messages, nodes chosen for relaying messages will also not be truly random, resulting in a possibility of using the same onion route more than once.</p>
Remediation	<ul style="list-style-type: none">- Seed <code>std::srand</code> during wallet initialisation- OR supply a random number generator to the <code>random_shuffle</code> function- OR use a safe randomization function instead
Details	Please refer to figure 9.4 on page 29 for evidence.



CLOAK

PROBLEM #3

Random generator used without Seed is resolved.

Vulnerability ID	4
Severity	Medium
Title	DLL Preloading attack / Hijacking
CVSS Vector	AV:L/AC:L/AU:N/C:P/I:C/A:P
CVSS Score	6.1
Assets	<ul style="list-style-type: none">• Cloak Application
Description	<p>cloakcoin-qt.exe tries to load system DLLs from the application's directory first, instead of searching from the system directory. This makes it easier for attackers to place additional files/dlls and to trick victims into running malicious code. For instance, the following dlls would be loaded from the application directory if they exist:</p> <ul style="list-style-type: none">- WINMM.DLL- IPHLPAPI.DLL- WINNSI.DLL- MSWSOCK.DLL
Remediation	<p>Consider removing the current directory from the standard search path by calling SetDllDirectory with an empty string ("").</p> <p>For more details, please refer to: https://msdn.microsoft.com/en-us/library/ff919712(VS.85).aspx</p>
Details	<p>Please refer to figure 9.5 on page 30 for evidence. Please refer to figure 9.6 on page 31 for evidence.</p>

PROBLEM #4, #5, #6, #7 and #8

These are all based on the use of the “outdated” bitcoin release and the system libraries used for the project. Detected DLL preloading vulnerabilities are actually OS dependent and will be mitigated by placing the needed DLLs in the protected system directories (e.g. system32 folder on Windows) when the production/release installer is created as a part of our new wallet package. Eventually, Cloak will be incorporated into source of the latest Litecoin project, also resolving many similar issues such as this that may arise.

Vulnerability ID	5
Severity	Medium
Title	DLLs without Address Space Layout Randomization (ASLR) and Data Execution Prevention (DEP) enabled
CVSS Vector	AV:A/AC:L/AU:N/C:N/I:C/A:N
CVSS Score	6.1
Assets	<ul style="list-style-type: none">• Cloak Application
Description	<p>The following third party DLLs are not compiled with ASLR and DEP enabled. This makes it easier for attackers to exploit vulnerabilities.</p> <ul style="list-style-type: none">- libcurl-4.dll- libeay32.dll- libevent-2-0-5.dll- libgcc_s_dw2-1.dll- libidn-1.1.dll- librtmp-1.dll- libssh2-1.dll- libstdc++-6.dll- libwinpthread-1.dll- ssleay32.dll
Remediation	The application should be compiled with ASLR and DEP enabled.
Details	<p>For more information, please refer to: https://blogs.technet.microsoft.com/srd/2014/03/12/when-aslr-makes-the-difference/ https://blogs.technet.microsoft.com/srd/2010/12/08/on-the-effectiveness-of-dep-and-aslr/</p> <p>Please refer to figure 9.7 on page 32 for evidence.</p>

PROBLEM #4, #5, #6, #7 and #8

These are all based on the use of the “outdated” bitcoin release and the system libraries used for the project. Detected DLL preloading vulnerabilities are actually OS dependent and will be mitigated by placing the needed DLLs in the protected system directories (e.g. system32 folder on Windows) when the production/release installer is created as a part of our new wallet package. Eventually, Cloak will be incorporated into source of the latest Litecoin project, also resolving many similar issues such as this that may arise.

Vulnerability ID	6
Severity	Medium
Title	Outdated Bitcoin code base (CVE-2013-2272, CVE-2013-4165, CVE-2013-4627)
CVSS Vector	AV:N/AC:L/Au:N/C:P/I:N/A:N
CVSS Score	5.0
Assets	<ul style="list-style-type: none">• Cloak Application

Vulnerability ID	6
Description	<p>The Cloak Coin application integrated an old version of the Bitcoin source code as underlying basis. This version of Bitcoin is outdated and has multiple potential vulnerabilities, for example:</p> <p>CVE-2013-2272 (Remote discovery of node's wallet addresses):</p> <p>The penny-flooding protection mechanism in the CTxMemPool::accept method in bitcoind and Bitcoin-Qt before 0.4.9rc1, 0.5.x before 0.5.8rc1, 0.6.0 before 0.6.0.11rc1, 0.6.1 through 0.6.5 before 0.6.5rc1, and 0.7.x before 0.7.3rc1 allows remote attackers to determine associations between wallet addresses and IP addresses via a series of large Bitcoin transactions with insufficient fees.</p> <p>CVE-2013-4165 (RPC password might be susceptible to timing attacks):</p> <p>The HTTPAuthorized function in bitcoinrpc.cpp in bitcoind 0.8.1 provides information about authentication failure upon detecting the first incorrect byte of a password, which makes it easier for remote attackers to determine passwords via a timing side-channel attack.</p> <p>CVE-2013-4627:</p> <p>Unspecified vulnerability in bitcoind and Bitcoin-Qt 0.8.x allows remote attackers to cause a denial of service (memory consumption) via a large amount of tx message data.</p> <p>Memory leak vulnerability:</p> <p>The function CKey::SignCompact contains a memory leak because the corresponding free() call is missing.</p> <p>Unsecure function memset():</p> <p>As memset() may not clean data completely and should not be used in privacy/security relevant code parts. OpenSSL provides the safe OPENSSL_cleanse() function in crypto.h, which is considered to be a safe alternative.</p>
Remediation	The Cloak Coin code should be adapted to use the latest version of the Bitcoin code base.

Vulnerability ID	6
Details	Please refer to figure 9.8 on page 33 for evidence. Please refer to figure 9.9 on page 34 for evidence.

PROBLEM #4, #5, #6, #7 and #8

These are all based on the use of the “outdated” bitcoin release and the system libraries used for the project. Detected DLL preloading vulnerabilities are actually OS dependent and will be mitigated by placing the needed DLLs in the protected system directories (e.g. system32 folder on Windows) when the production/release installer is created as a part of our new wallet package. Eventually, Cloak will be incorporated into source of the latest Litecoin project, also resolving many similar issues such as this that may arise.

Vulnerability ID	7
Severity	Medium
Title	Outdated Tor code base (CVE-2017-8819, CVE-2017-8821, CVE-2017-8822, CVE-2017-0375, CVE-2017-0376, CVE-2016-1254)
CVSS Vector	AV:N/AC:L/Au:N/C:P/I:N/A:N
CVSS Score	5.0
Assets	<ul style="list-style-type: none">• Cloak Application
Description	The Cloak application integrated an old version of the Tor source code as the underlying code for Cloakshield (0.2.5.1-alpha-dev).
Remediation	The Cloak code should be adapted to use the latest version of the Tor code base. Alternatively, fixes for known vulnerabilities could be backported.

Vulnerability ID	7
Details	<p>The used version of Tor is outdated and has multiple potential vulnerabilities, for example:</p> <p>CVE-2017-8819 (Limited replay attack of INTRODUCE2 cells):</p> <p>In Tor before 0.2.5.16, 0.2.6 through 0.2.8 before 0.2.8.17, 0.2.9 before 0.2.9.14, 0.3.0 before 0.3.0.13, and 0.3.1 before 0.3.1.9, the replay-cache protection mechanism is ineffective for v2 onion services, aka TROVE-2017-009. An attacker can send many INTRODUCE2 cells to trigger this issue.</p> <p>CVE-2017-8821 (An attacker can make Tor ask for a password):</p> <p>In Tor before 0.2.5.16, 0.2.6 through 0.2.8 before 0.2.8.17, 0.2.9 before 0.2.9.14, 0.3.0 before 0.3.0.13, and 0.3.1 before 0.3.1.9, an attacker can cause a denial of service (application hang) via crafted PEM input that signifies a public key requiring a password, which triggers an attempt by the OpenSSL library to ask the user for the password, aka TROVE-2017-011.</p> <p>CVE-2017-8822 (Relays can pick themselves in a circuit path):</p> <p>In Tor before 0.2.5.16, 0.2.6 through 0.2.8 before 0.2.8.17, 0.2.9 before 0.2.9.14, 0.3.0 before 0.3.0.13, and 0.3.1 before 0.3.1.9, relays (that have incompletely downloaded descriptors) can pick themselves in a circuit path, leading to a degradation of anonymity, aka TROVE-2017-012.</p> <p>CVE-2017-0375 & CVE-2017-0376 (Assertion failure and daemon exit):</p> <p>The hidden-service feature in Tor before 0.3.0.8 allows a denial of service (assertion failure and daemon exit) in the relay_send_end_cell_from_edge_function via a malformed BEGIN cell.</p> <p>CVE-2016-1254 (Out of bounds read):</p> <p>Tor before 0.2.8.12 might allow remote attackers to cause a denial of service (client crash) via a crafted hidden service descriptor.</p>

PROBLEM #4, #5, #6, #7 and #8

These are all based on the use of the “outdated” bitcoin release and the system libraries used for the project. Detected DLL preloading vulnerabilities are actually OS dependent and will be mitigated by placing the needed DLLs in the protected system directories (e.g. system32 folder on Windows) when the production/release installer is created as a part of our new wallet package. Eventually, Cloak will be incorporated into source of the latest Litecoin project, also resolving many similar issues such as this that may arise.

Vulnerability ID	8
Severity	Medium
Title	Static Source Code Analysis
CVSS Vector	AV:L/AC:L/AU:N/C:P/I:P/A:P
CVSS Score	4.6
Assets	<ul style="list-style-type: none">• Cloak Application
Description	<p>The application uses of functions that are potentially dangerous and are not recommended to be used anymore as they might leave the application vulnerable to e.g. buffer overflows. As a consequence, attackers might be able to target other wallets.</p> <p>For instance, the following potentially dangerous functions could be found in the source code:</p> <p>scanf memcpy goto sprintf</p>
Remediation	The potentially vulnerable functions should not be used.
Details	Please refer to the provided Excel file for the detailed findings.

PROBLEM #4, #5, #6, #7 and #8

These are all based on the use of the “outdated” bitcoin release and the system libraries used for the project. Detected DLL preloading vulnerabilities are actually OS dependent and will be mitigated by placing the needed DLLs in the protected system directories (e.g. system32 folder on Windows) when the production/release installer is created as a part of our new wallet package. Eventually, Cloak will be incorporated into source of the latest Litecoin project, also resolving many similar issues such as this that may arise.

Vulnerability ID	9
Severity	Medium
Title	Flawed Splitting Randomizer
CVSS Vector	AV:N/AC:M/AU:N/C:P/I:N/A:N
CVSS Score	4.3
Assets	<ul style="list-style-type: none">• Cloak Application
Description	<p>Per design, each transaction input should be split into either two or three outputs. The randomizer function, GetRandRange, did not return a randomized integer, but instead the value 2 was returned at all times with the supplied parameters.</p> <p>This resulted in output addresses to be always split by 2, which did not lead to the expected level of anonymity.</p>
Remediation	The GetRandRange function should be improved to return randomized values only.

Vulnerability ID	9
Details	<p>Per design each input transaction should be split to either two or three outputs. This decision was randomized by the function <code>GetRandRange(int64 nMin, int64 nMax)</code>, which called the inherited <code>GetRand(uint64 nMax)</code> function from the original Bitcoin code.</p> <p>The function <code>GetRandRange(int64 nMin, int64 nMax)</code> got the following values:</p> <p><code>nMin = 2</code> <code>nMax = 3</code></p> <p>The idea was to get a random value of 2 or 3.</p> <p>Unfortunately the <code>GetRand</code> function was called incorrectly using following function call:</p> <pre>return GetRand(nMax - nMin) + nMax;</pre> <p>This lead to the function <code>GetRand(uint64 nMax)</code> to be called with the value of 1, which was used to calculate a random value by using the modulo operator:</p> <pre>return (nRand % nMax);</pre> <p>As the value 1 was supplied to <code>nMax</code>, the result will always be 0.</p> <p>Going back to the inherited call</p> <pre>return GetRand(nMax - nMin) + nMax;</pre> <p>and replacing the variables with actual values:</p> <pre>return GetRand(0) + 2;</pre> <p>we can see that there was in fact no randomization done by the function and every transaction is split into exactly 2 outputs.</p> <p>Please refer to figure 9.10 on page 35 for evidence. Please refer to figure 9.11 on page 36 for evidence. Please refer to figure 9.12 on page 36 for evidence.</p>



CLOAK

PROBLEM #9

Flawed Splitting Randomizer is resolved as mentioned in Problem #1 and discussed in the appendix.

Vulnerability ID	10
Severity	Low
Title	Weak Backup Methods
CVSS Vector	AV:L/AC:L/Au:N/C:N/I:N/A:P
CVSS Score	2.1
Assets	<ul style="list-style-type: none">• Cloak Application
Description	<p>According to the Cryptocurrency Security Standard, the wallet key should be backed up via a seed phrase. This makes it possible to store the recovery phrase on paper and non-digital devices.</p> <p>In this case, no seed phrase is defined on wallet creation. The backup is done via copying the wallet.dat file to another location. Using a seed phrase would make it easier for users to backup their wallet on non-digital devices.</p>
Remediation	A system to backup the wallet via seed phrases should be implemented to ensure backup and recovery via non-digital devices.



CLOAK

PROBLEM #10

Weak Backup Methods is under consideration for future releases.

Vulnerability ID	11
Severity	Informational
Title	Incorrect Number of Cloakers used
CVSS Vector	N/A
CVSS Score	0.0
Assets	<ul style="list-style-type: none">• Cloak Application
Description	<p>When sending an Enigma transaction, it was discovered that one cloaker less then selected by the user was actually used to cloak the transaction. This decreases the anonymity of the transaction, especially because users are not aware of this fact.</p> <p>For example, if the user selects to use 3 cloakers, only 2 are actually used to cloak the transaction.</p> <p>According to the customer, that represents intended behaviour as the sender is technically also considered a cloaker. Therefore the severity of that finding has been reduced to informational.</p>
Remediation	The number of cloakers which is selected by the user should be used. Users should be informed that a sender is also considered a cloaker.
Details	<p>Please refer to figure 9.13 on page 37 for evidence.</p> <p>Please refer to figure 9.14 on page 38 for evidence.</p>

PROBLEM #11

Incorrect Number of Cloakers used is, as Cognosec determined, informational. Cognosec failed to recognize the sender as a participant of the Enigma transaction. When looking through their analysis, which provides code, it can be seen that the number of participants in the software is reduced by 1, specifically. The highlighted line in Figure 9.13, of the audit, uses the function `CreateForBroadcast` with the first parameter, `numParticipants - 1`. This is also described in the appendix where 4 Cloakers are used. On live net, the current minimum number of Cloakers is set to 5 and can be set by the user. This enhances the difficulty of analyzing the transactions even further.

8 ABBREVIATIONS

ASV	Approved Scanning Vendor
PCI DSS	Payment Card Industrie (PCI) Data Security Standard (DSS)
PCI SSC	PCI Security Standards Council, LLC.
CISA	Certified Information Systems Auditor
CISSP	Certified Information System Security Professional
PIN	Personal Identification Number
IVR	Interactive Voice Response
CVSS	Common Vulnerability Scoring System
NVD	National Vulnerability Database
CGI	Common Gateway Interface
CSRF	Cross-site request forgery
DNS	Domain Name System
DSS	Data Security Standard
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ISACA	Information Systems Audit and Control Association
ITAF	Information Technology Assurance Framework
LDAP	Lightweight Directory Access Protocol
OSSTMM	Open Source Security Testing Methodology Manual
OWASP	Open Web Application Security Project
RPC	Remote Procedure Call
SCADA	Supervisory Control And Data Acquisition
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SSL	Secure Sockets Layer
SQL	Structured Query Language
TLS	Transport Layer Security
URL	Uniform Resource Locator
XSS	Cross-site scripting

9 APPENDIX

9.1 Evidence

9.1.1 Evidence for Vulnerability ID 1: Compromise of Anonymity

Address	Amount	Identified Type	
mt1gdL9PEfMAuxecgm7GTGdhwDHJ5dXSEN	0,796567	Output of cloaker or recipient	
mkNgRAGD1wETk8QVjNgzvDH9m4oHn4a9Kj	90,153015	Sender credit	
mfiBiyUEeS2gNEDJ6NztR4bQ96JMHbDnha	395,626985	Sender credit	
mr33vFwzWnVLG5VjG9jnfMCvCwbX3SgfuE	6,767247	Output of cloaker or recipient	
mrLH4ET6pxQdJoVRG8SgWPWM5KNxkoaDr	0,575900	Output of cloaker or recipient	
mJPmo4A7CVbDLqxoNrHvLSMDy6VXYVvSjf	0,049757	Split of change + reward	
mwog94chutVZwRu68zZm1s1PpcYsEd5wde	9,424100	Output of cloaker or recipient	
mhm79gNPGLVajMbwpYqw1evG1VV1YYMgB8	9,203433	Output of cloaker or recipient	
mnbDxmZbZagmbbFg9qBn5wLvHPUxsjnEhW	0,111599	Split of change + reward	
mxsAYBBdTNC7VBhLYag6nKrNvsMg7N3vvH	0,067248	Split of change + reward	
mkksfA5q2km97AsbwnQoY33TPe5KZVWCUJ	3,232753	Output of cloaker or recipient	
	516,008604		
Actual Transaction Amount:	10,000000		
		Sum Yellow:	10,000000
		Sum Red:	10,000000
		Sum Green:	10,000000
		Credit for Sender, similar to Change + Reward	485,780000
		Sum Change + Reward	0,228604
		Sum Change without Reward	0,048604

Figure 9.1: By building the sum of combinations of the output amounts, the value with the most occurrences represents the transferred amount of Cloaks -> 10 Cloaks in this case. That way three parties (recipient + cloakers) were identified.

	A	B
1	Combinations of change and reward outputs	Transaction amount minus reward (0,9% for each cloaker, as two are used)
2	0,049757	9,959757
3	0,111599	10,021599
4	0,067248	9,977248
5	0,161356	10,071356
6	0,117005	10,027005
7	0,178847	10,088847
8		
9		
10		Red is the matched value in the combinations of the inputs
11		

Figure 9.2: Combinatons of residual change+reward outputs are built and compared to the sum of combinations of the input amounts. That way on the one hand the inputs of the cloakers and on the other hand the input of the sender can be determined (highlighted red in that case).

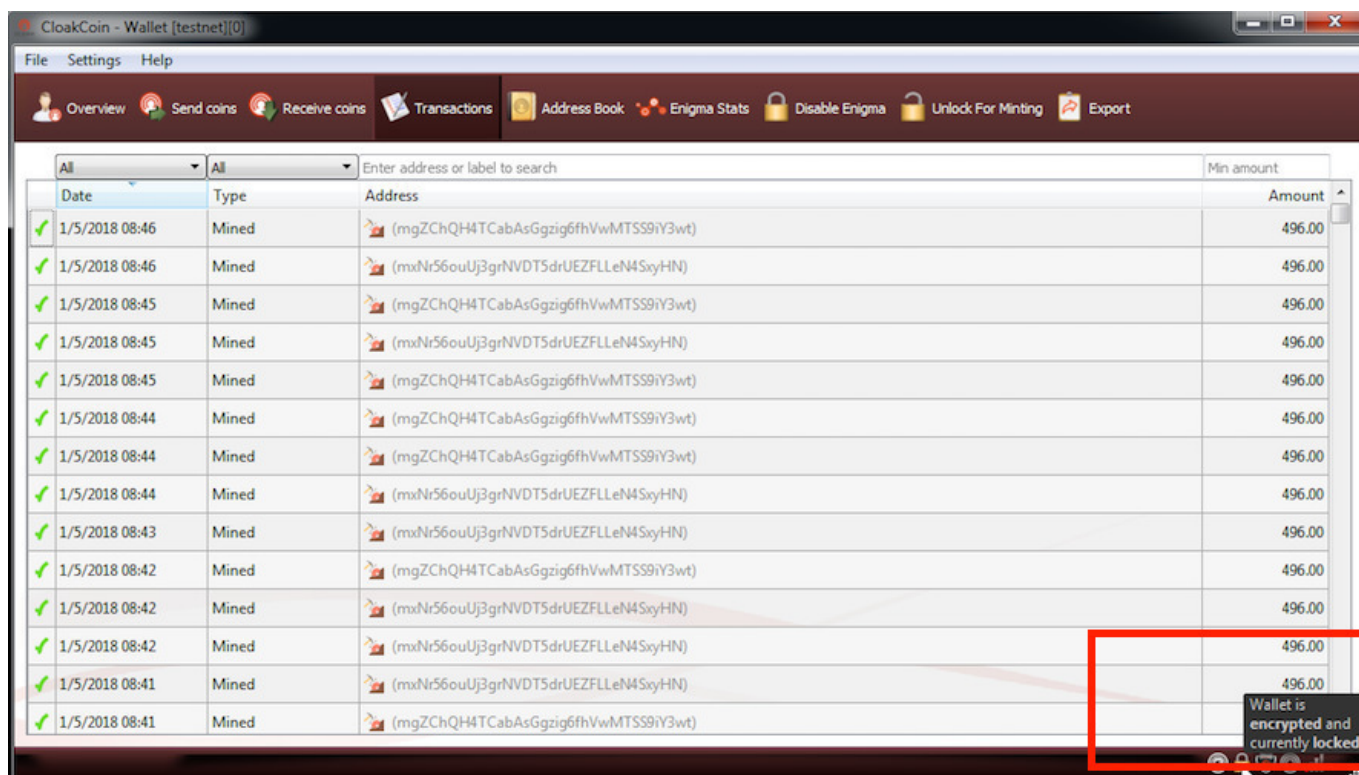
APPENDIX

Recipients Address	(CLOAK) Amount	Participant 1	Participant 2	Participant 3	Participant 4
mryjdnzyQvoQY7W2r48	0,066728				
mjrok5827gsd8RKxvEgB	435,017429	435,017429			
muJ38R79zAX93ngziCB	14,566064	14,566064			
n1Z5VpjBAbUGAGG85	46,004393		46,004393		
mnurFmpqLz9Y8qf9Yp	421,453968				421,453968
mg6i3y8mAwPsihE9Hi5	283,439695		283,439695		
miWYJbVxZmyNkHFs9	0,202846				
mmQZvVyqRskKX7Apt	179,214218		179,214218		
mssedFiRmDaas1sYQkD	88,759484			88,759484	
mjmA8QPwRKaDCS1R	0,094191				
mw4JtfQFTZoiQr9dpRB	602,334454			602,334454	
mqwMnKmJNVYPaxE8	307,58621	307,58621			
n4MydoefnrDEEtNGLTS	314,190202			314,190202	
mm76shzWtvj3H8PWHp	249,90514	249,90514			
mj2zCafKDibCisQPNVw	171,480379				171,480379
mjGV3ZUxCYshFwBkB	498,194921		498,194921		
mtkfu25dHmWrx1TZuc	407,065653				407,065653
	4019,57598	1007,07484	1006,85323	1005,28414	1000

Shown above is a typical transaction of 1000 CloakCoins sent with Enigma, four Cloakers and Enigma fees of 1.8%. This enhancement prevents using possibility analysis to determine the sent amount. And results in preventing the ability to find the sender. From this transaction, it is also not possible to determine the amount by attempting to reverse engineer a 1.8% Enigma fee on any transaction since these values do not provide any information leading to such a conclusion.

Also, clearly shown, is the solution to Problem #9. In this transaction of 1000 CloakCoins, Participant 1 split their transaction four times, Participant 2 four times, Participant 3 three times and Participant 4 three times. Without direct knowledge of how many participants are involved or how many splits are applied to each individual participant, it appears impossible to determine the transaction amount and which transactions should be applied to the sender.

9.1.2 Evidence for Vulnerability ID 2: Insufficient Wallet Encryption



Date	Type	Address	Amount
1/5/2018 08:46	Mined	(mgZChQH4TCabAsGgzig6fhVwMTSS9iY3wt)	496.00
1/5/2018 08:46	Mined	(mxNr56ouUj3grNVDTSdrUEZFLLeN4SxyHN)	496.00
1/5/2018 08:45	Mined	(mgZChQH4TCabAsGgzig6fhVwMTSS9iY3wt)	496.00
1/5/2018 08:45	Mined	(mxNr56ouUj3grNVDTSdrUEZFLLeN4SxyHN)	496.00
1/5/2018 08:45	Mined	(mgZChQH4TCabAsGgzig6fhVwMTSS9iY3wt)	496.00
1/5/2018 08:44	Mined	(mgZChQH4TCabAsGgzig6fhVwMTSS9iY3wt)	496.00
1/5/2018 08:44	Mined	(mgZChQH4TCabAsGgzig6fhVwMTSS9iY3wt)	496.00
1/5/2018 08:44	Mined	(mxNr56ouUj3grNVDTSdrUEZFLLeN4SxyHN)	496.00
1/5/2018 08:43	Mined	(mxNr56ouUj3grNVDTSdrUEZFLLeN4SxyHN)	496.00
1/5/2018 08:42	Mined	(mgZChQH4TCabAsGgzig6fhVwMTSS9iY3wt)	496.00
1/5/2018 08:42	Mined	(mxNr56ouUj3grNVDTSdrUEZFLLeN4SxyHN)	496.00
1/5/2018 08:42	Mined	(mxNr56ouUj3grNVDTSdrUEZFLLeN4SxyHN)	496.00
1/5/2018 08:41	Mined	(mxNr56ouUj3grNVDTSdrUEZFLLeN4SxyHN)	496.00
1/5/2018 08:41	Mined	(mgZChQH4TCabAsGgzig6fhVwMTSS9iY3wt)	496.00

Figure 9.3: Even though the wallet is encrypted, the transactions are visible.

9.1.3 Evidence for Vulnerability ID 3: Random generator used without seed

```
....
2229
2230 // -- shuffle inputs, change output won't mix enough as it must be not fully random for plaintext narrations
2231: std::random_shuffle(vecSend.begin(), vecSend.end());
2232
2233 int nChangePos;

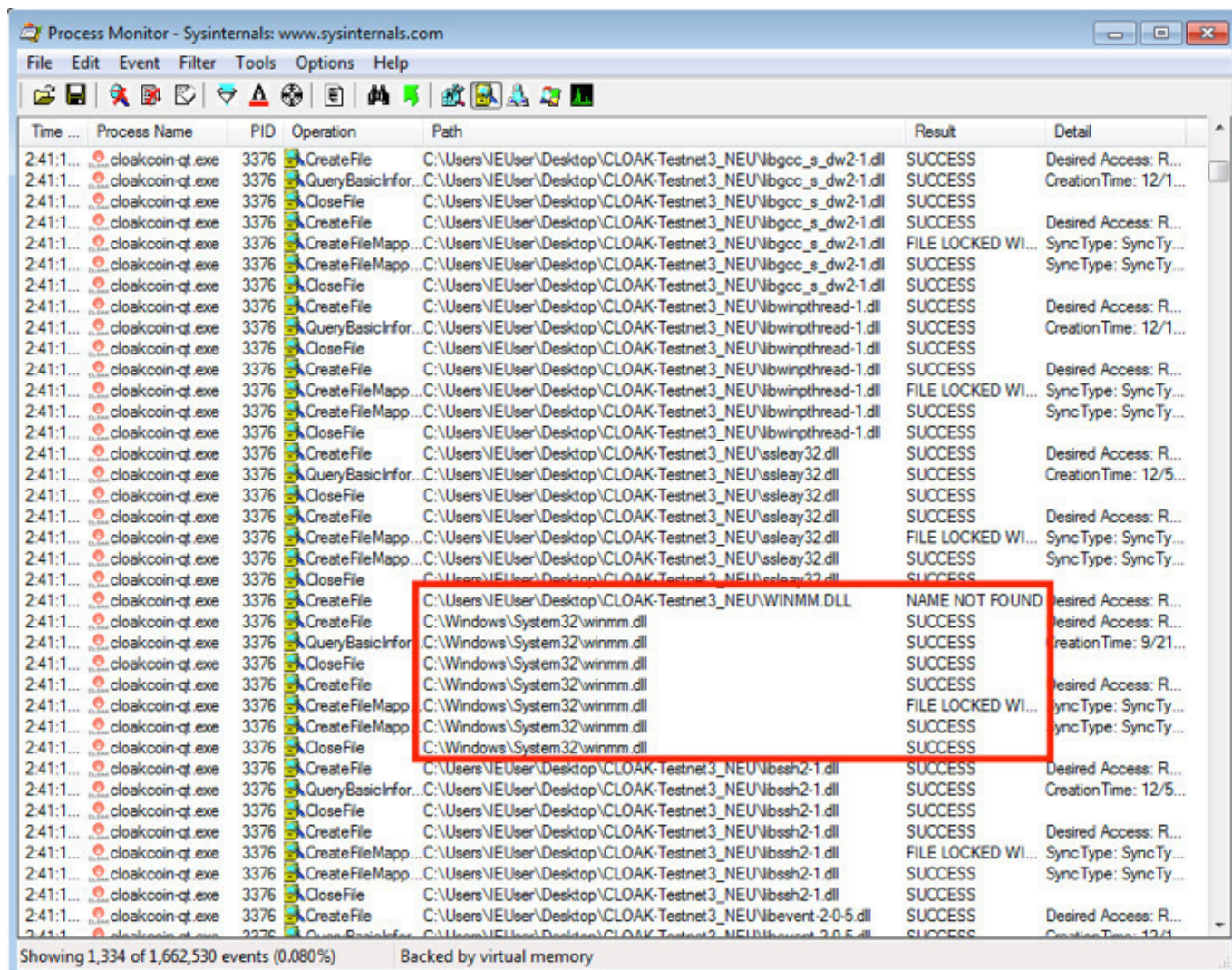
                                /Cloakcoin/Cloak2-2.1.0/src/enigma/cloakingdata.cpp:
135     {
136         // shuffle available nodes so we end up with a different subset per route
137:         std::random_shuffle(relayNodes.begin(), relayNodes.end());
138
139         vector<CEnigmaAnnouncement> routeNodes;
140
141         {
142             // shuffle available nodes so we end up with a different subset per hop
143:             std::random_shuffle(routeNodes.begin(), routeNodes.end());
144
145             // encode this data in wrapper data for target node
146
147         }
148
149         if (shuffle)
150:             std::random_shuffle(shuffledNodes.begin(), shuffledNodes.end());
151
152         CCloakingData datapacket = packetsToSend[i];

                                /Cloakcoin/Cloak2-2.1.0/src/enigma/cloakingrequest.cpp:
390
391         // shuffle inputs and outputs
392:         random_shuffle(inOuts.vin.begin(), inOuts.vin.end());
393:         random_shuffle(inOuts.vout.begin(), inOuts.vout.end());
394
395         // create zero output with op return and nonce for stealth change collection

                                /Cloakcoin/Cloak2-2.1.0/src/enigma/enigmaann.cpp:
233     {
234         // shuffle
235:         std::random_shuffle(enigmaAnnsAvail.begin(), enigmaAnnsAvail.end());
236
237         // crop to size
```

Figure 9.4: The random_shuffle function was used 7 times without supplying a random generator (which would be the third argument)

9.1.4 Evidence for Vulnerability ID 4: DLL Preloading attack / Hijacking



Time ...	Process Name	PID	Operation	Path	Result	Detail
2:41:1...	cloakcoin-qt.exe	3376	CreateFile	C:\Users\VEUser\Desktop\CLOAK-Testnet3_NEU\Nbicc_s_dw2-1.dll	SUCCESS	Desired Access: R...
2:41:1...	cloakcoin-qt.exe	3376	QueryBasicInfo...	C:\Users\VEUser\Desktop\CLOAK-Testnet3_NEU\Nbicc_s_dw2-1.dll	SUCCESS	CreationTime: 12/1...
2:41:1...	cloakcoin-qt.exe	3376	CloseFile	C:\Users\VEUser\Desktop\CLOAK-Testnet3_NEU\Nbicc_s_dw2-1.dll	SUCCESS	
2:41:1...	cloakcoin-qt.exe	3376	CreateFile	C:\Users\VEUser\Desktop\CLOAK-Testnet3_NEU\Nbicc_s_dw2-1.dll	SUCCESS	Desired Access: R...
2:41:1...	cloakcoin-qt.exe	3376	CreateFileMapp...	C:\Users\VEUser\Desktop\CLOAK-Testnet3_NEU\Nbicc_s_dw2-1.dll	FILE LOCKED WI...	SyncType: SyncTy...
2:41:1...	cloakcoin-qt.exe	3376	CreateFileMapp...	C:\Users\VEUser\Desktop\CLOAK-Testnet3_NEU\Nbicc_s_dw2-1.dll	SUCCESS	SyncType: SyncTy...
2:41:1...	cloakcoin-qt.exe	3376	CreateFile	C:\Users\VEUser\Desktop\CLOAK-Testnet3_NEU\Nbwinpthread-1.dll	SUCCESS	Desired Access: R...
2:41:1...	cloakcoin-qt.exe	3376	QueryBasicInfo...	C:\Users\VEUser\Desktop\CLOAK-Testnet3_NEU\Nbwinpthread-1.dll	SUCCESS	CreationTime: 12/1...
2:41:1...	cloakcoin-qt.exe	3376	CloseFile	C:\Users\VEUser\Desktop\CLOAK-Testnet3_NEU\Nbwinpthread-1.dll	SUCCESS	
2:41:1...	cloakcoin-qt.exe	3376	CreateFile	C:\Users\VEUser\Desktop\CLOAK-Testnet3_NEU\Nbwinpthread-1.dll	SUCCESS	Desired Access: R...
2:41:1...	cloakcoin-qt.exe	3376	CreateFileMapp...	C:\Users\VEUser\Desktop\CLOAK-Testnet3_NEU\Nbwinpthread-1.dll	FILE LOCKED WI...	SyncType: SyncTy...
2:41:1...	cloakcoin-qt.exe	3376	CreateFileMapp...	C:\Users\VEUser\Desktop\CLOAK-Testnet3_NEU\Nbwinpthread-1.dll	SUCCESS	SyncType: SyncTy...
2:41:1...	cloakcoin-qt.exe	3376	CreateFile	C:\Users\VEUser\Desktop\CLOAK-Testnet3_NEU\ssleay32.dll	SUCCESS	Desired Access: R...
2:41:1...	cloakcoin-qt.exe	3376	QueryBasicInfo...	C:\Users\VEUser\Desktop\CLOAK-Testnet3_NEU\ssleay32.dll	SUCCESS	CreationTime: 12/5...
2:41:1...	cloakcoin-qt.exe	3376	CloseFile	C:\Users\VEUser\Desktop\CLOAK-Testnet3_NEU\ssleay32.dll	SUCCESS	
2:41:1...	cloakcoin-qt.exe	3376	CreateFile	C:\Users\VEUser\Desktop\CLOAK-Testnet3_NEU\ssleay32.dll	SUCCESS	Desired Access: R...
2:41:1...	cloakcoin-qt.exe	3376	CreateFileMapp...	C:\Users\VEUser\Desktop\CLOAK-Testnet3_NEU\ssleay32.dll	FILE LOCKED WI...	SyncType: SyncTy...
2:41:1...	cloakcoin-qt.exe	3376	CreateFileMapp...	C:\Users\VEUser\Desktop\CLOAK-Testnet3_NEU\ssleay32.dll	SUCCESS	SyncType: SyncTy...
2:41:1...	cloakcoin-qt.exe	3376	CloseFile	C:\Users\VEUser\Desktop\CLOAK-Testnet3_NEU\ssleay32.dll	SUCCESS	
2:41:1...	cloakcoin-qt.exe	3376	CreateFile	C:\Users\VEUser\Desktop\CLOAK-Testnet3_NEU\WINMM.DLL	NAME NOT FOUND	Desired Access: R...
2:41:1...	cloakcoin-qt.exe	3376	CreateFile	C:\Windows\System32\winmm.dll	SUCCESS	Desired Access: R...
2:41:1...	cloakcoin-qt.exe	3376	QueryBasicInfo...	C:\Windows\System32\winmm.dll	SUCCESS	CreationTime: 9/21...
2:41:1...	cloakcoin-qt.exe	3376	CloseFile	C:\Windows\System32\winmm.dll	SUCCESS	
2:41:1...	cloakcoin-qt.exe	3376	CreateFile	C:\Windows\System32\winmm.dll	SUCCESS	Desired Access: R...
2:41:1...	cloakcoin-qt.exe	3376	CreateFileMapp...	C:\Windows\System32\winmm.dll	FILE LOCKED WI...	SyncType: SyncTy...
2:41:1...	cloakcoin-qt.exe	3376	CreateFileMapp...	C:\Windows\System32\winmm.dll	SUCCESS	SyncType: SyncTy...
2:41:1...	cloakcoin-qt.exe	3376	CloseFile	C:\Windows\System32\winmm.dll	SUCCESS	
2:41:1...	cloakcoin-qt.exe	3376	CreateFile	C:\Users\VEUser\Desktop\CLOAK-Testnet3_NEU\Nbssh2-1.dll	SUCCESS	Desired Access: R...
2:41:1...	cloakcoin-qt.exe	3376	QueryBasicInfo...	C:\Users\VEUser\Desktop\CLOAK-Testnet3_NEU\Nbssh2-1.dll	SUCCESS	CreationTime: 12/5...
2:41:1...	cloakcoin-qt.exe	3376	CloseFile	C:\Users\VEUser\Desktop\CLOAK-Testnet3_NEU\Nbssh2-1.dll	SUCCESS	
2:41:1...	cloakcoin-qt.exe	3376	CreateFile	C:\Users\VEUser\Desktop\CLOAK-Testnet3_NEU\Nbssh2-1.dll	SUCCESS	Desired Access: R...
2:41:1...	cloakcoin-qt.exe	3376	CreateFileMapp...	C:\Users\VEUser\Desktop\CLOAK-Testnet3_NEU\Nbssh2-1.dll	FILE LOCKED WI...	SyncType: SyncTy...
2:41:1...	cloakcoin-qt.exe	3376	CreateFileMapp...	C:\Users\VEUser\Desktop\CLOAK-Testnet3_NEU\Nbssh2-1.dll	SUCCESS	SyncType: SyncTy...
2:41:1...	cloakcoin-qt.exe	3376	CloseFile	C:\Users\VEUser\Desktop\CLOAK-Testnet3_NEU\Nbssh2-1.dll	SUCCESS	
2:41:1...	cloakcoin-qt.exe	3376	CreateFile	C:\Users\VEUser\Desktop\CLOAK-Testnet3_NEU\Nbevent-2-0-5.dll	SUCCESS	Desired Access: R...
2:41:1...	cloakcoin-qt.exe	3376	QueryBasicInfo...	C:\Users\VEUser\Desktop\CLOAK-Testnet3_NEU\Nbevent-2-0-5.dll	SUCCESS	CreationTime: 12/1...

Showing 1,334 of 1,662,530 events (0.080%) Backed by virtual memory

Figure 9.5: The highlighted system calls show that the application tries to load the file winmm.dll from the applications directory before looking in the System32 directory.

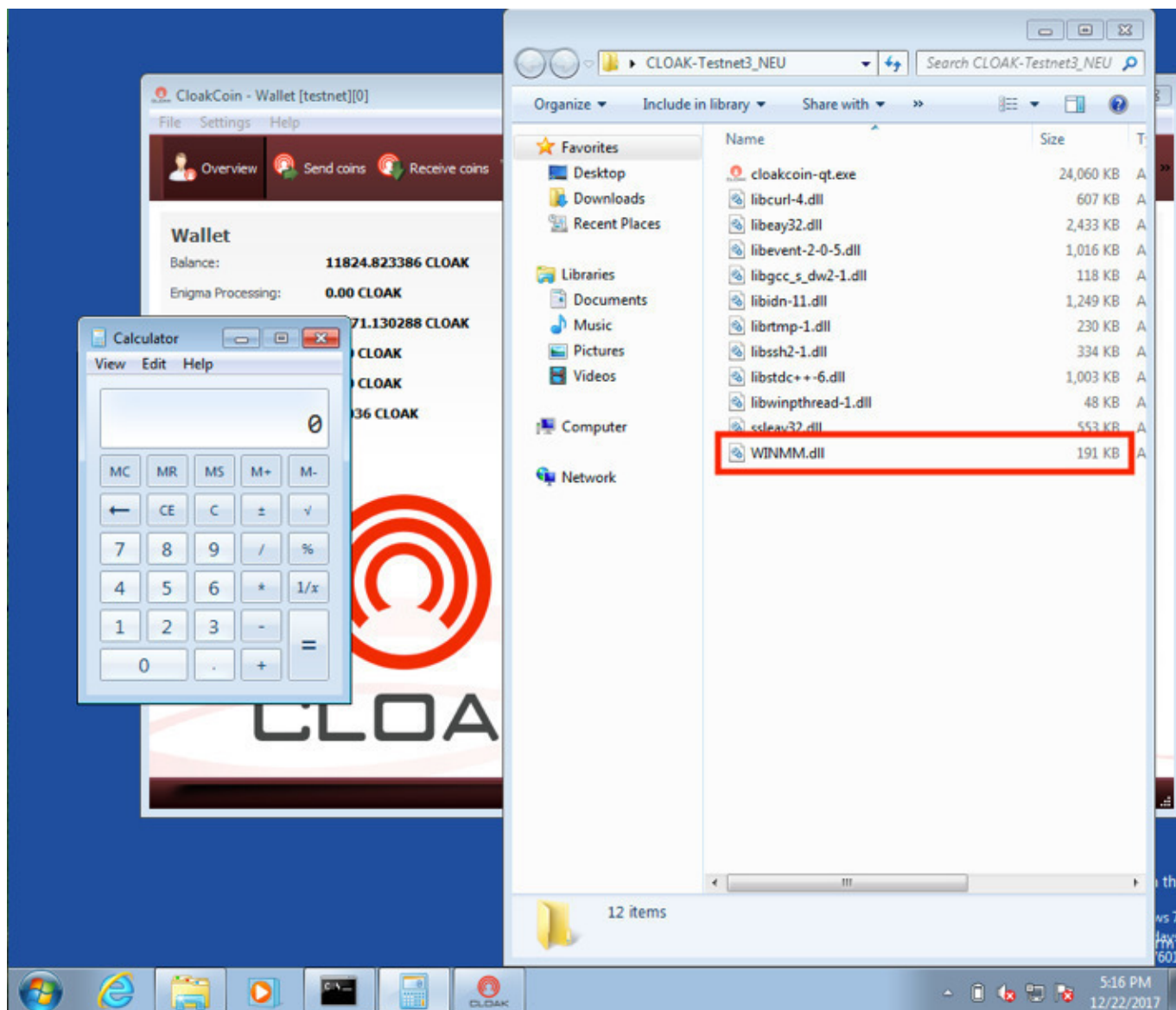


Figure 9.6: By copying a modified version of that file in the application's folder, it was possible to trigger the execution of additional applications (in this example, the calculator).

9.1.5 Evidence for Vulnerability ID 5: DLLs without Address Space Layout Randomization (ASLR) and Data Execution Prevention (DEP) enabled

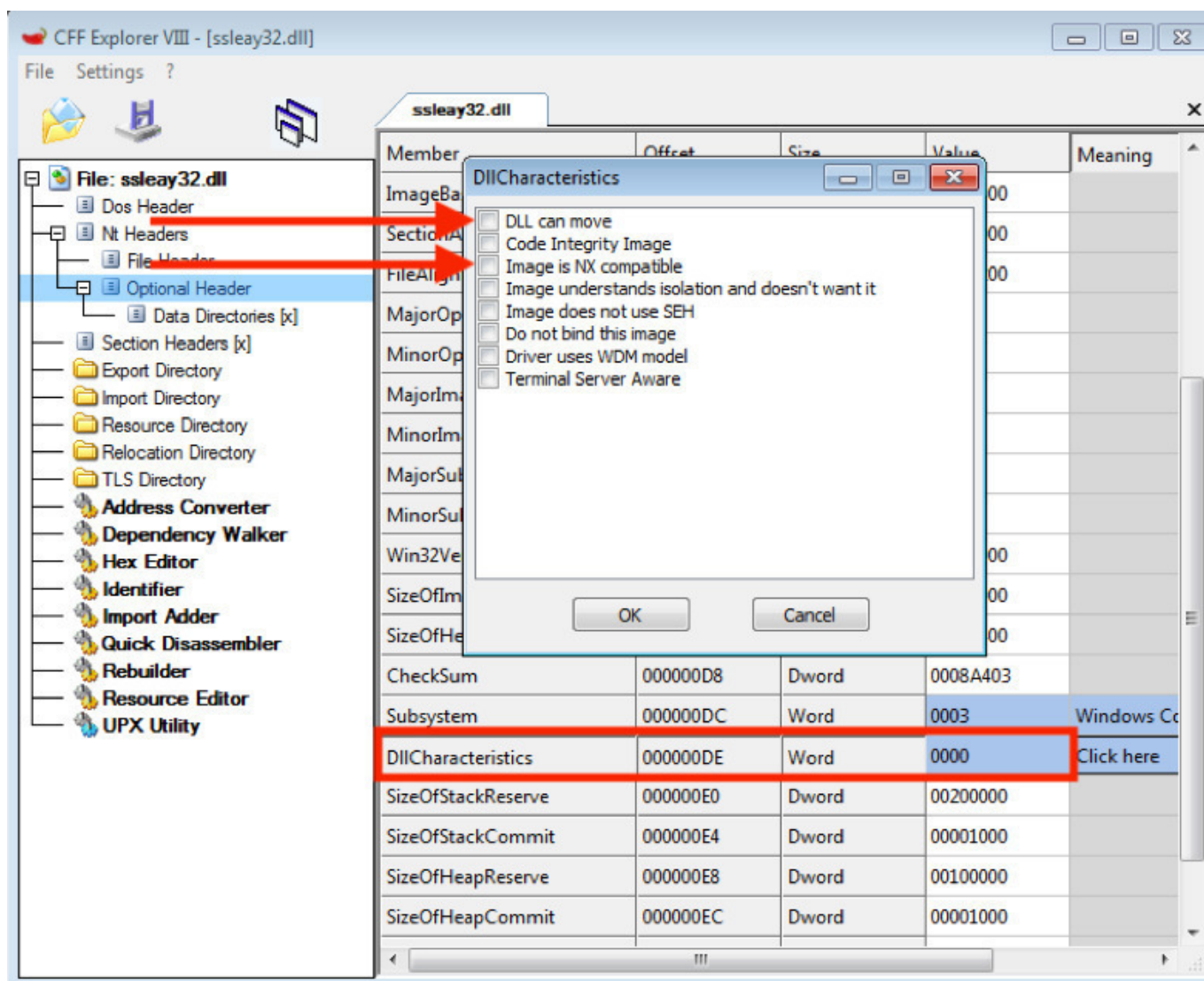


Figure 9.7: This caption shows the options used for compilation.

9.1.6 Evidence for Vulnerability ID 6: Outdated Bitcoin code base (CVE-2013-2272, CVE-2013-4165, CVE-2013-4627)

```
578     if (nBlockSize != 1 && nNewBlockSize >= MAX_BLOCK_SIZE_GEN/2)
579     {
580         if (nNewBlockSize >= MAX_BLOCK_SIZE_GEN)
581             return MAX_MONEY;
582         nMinFee *= MAX_BLOCK_SIZE_GEN / (MAX_BLOCK_SIZE_GEN - nNewBlockSize);
583     }
584
585     if (!MoneyRange(nMinFee))
586         nMinFee = MAX_MONEY;
587     return nMinFee;
588 }
589
590
591 bool CTxMemPool::accept(CTxDB& txdb, CTransaction &tx, bool fCheckInputs,
592                         bool* pfMissingInputs)
593 {
594     if (pfMissingInputs)
595         *pfMissingInputs = false;
596
597     if (!tx.CheckTransaction())
598         return error("CTxMemPool::accept() : CheckTransaction failed");
599
600     // Coinbase is only valid in a block, not as a loose transaction
601     if (tx.IsCoinBase())
```

Figure 9.8: This is an example of the outdated Bitcoin code within the Cloak Coin source code.

Remove IsFromMe() check in CTxMemPool::accept()

Fixes issue #2178 : attacker could penny-flood with invalid-signature transactions to deduce which addresses belonged to your node.


I'm committing this early for code review; I still need to write up a test plan.

Executive summary of fix: check all transactions received from the network for penny-flood rate-limiting before adding to the memory pool. But do NOT ratelimit transactions added to the memory pool:

- because of blockchain reorgs
- stored in the wallet and added at startup
- sent from the GUI or one of the send* RPC commands (CWallet::CommitTransaction)

The limit-free-transactions code really should be a method on CNode, with counters per-peer. But that is a bigger change for another day.

master (#2182) v0.15.1 v0.8.0rc1

 gavinandresen committed on 14 Jan 2013 1 parent c83c3cb commit ce99358f4aa4182d6983fde3e33a8fdb1dfe4c3

Showing 4 changed files with 31 additions and 32 deletions.

53 src/main.cpp View

```

@@ -627,7 +627,7 @@ void CTxMemPool::pruneSpent(const uint256 &hashTx, CCoins &coins)
627 627     }
628 628     }
629 629
630 -bool CTxMemPool::accept(CTransaction &tx, bool fCheckInputs,
630 +bool CTxMemPool::accept(CTransaction &tx, bool fCheckInputs, bool fLimitFree,
631 631                             bool* pfMissingInputs)
632 632     {

```

[Browse files](#)

Figure 9.9: The Bitcoin Git changelog shows the vulnerable code and the applied fix.

9.1.7 Evidence for Vulnerability ID 9: Flawed Splitting Randomizer

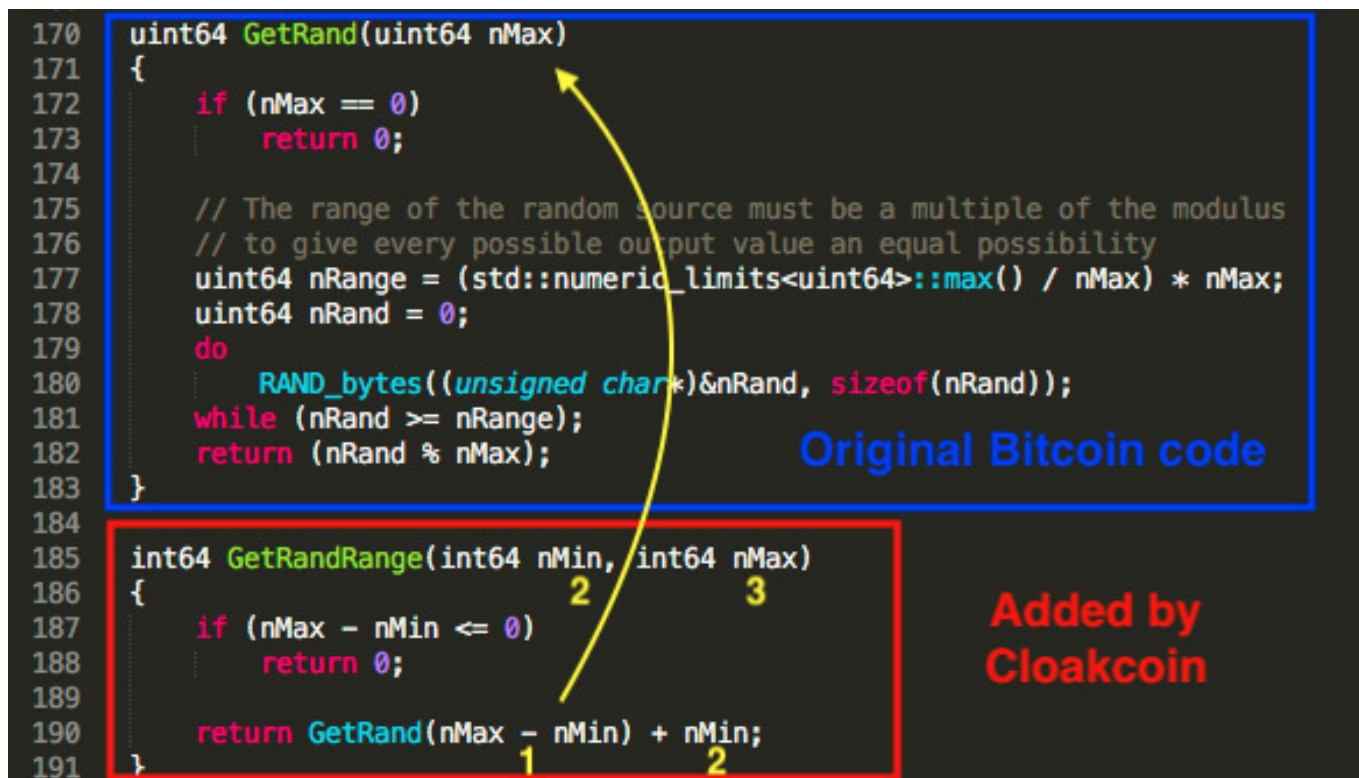


Figure 9.10: The idea was to get a random value between 2 and 3, the GetRand function was called with the value 1.


```

170 uint64 GetRand(uint64 nMax)
171 {
172     if (nMax == 0)
173         return 0;
174
175     // The range of the random source must be a multiple of the modulus
176     // to give every possible output value an equal possibility
177     uint64 nRange = (std::numeric_limits<uint64>::max() / nMax) * nMax;
178     uint64 nRand = 0;
179     do
180         RAND_bytes((unsigned char*)&nRand, sizeof(nRand));
181     while (nRand >= nRange);
182     return (nRand % nMax);
183 }
184
185 int64 GetRandRange(int64 nMin, int64 nMax)
186 {
187     if (nMax - nMin <= 0)
188         return 0;
189
190     return GetRand(nMax - nMin) + nMin;
191 }

```

Original Bitcoin code

Added by Cloakcoin

If nMax = 1, (nRand % 1) = 0

Figure 9.11: If 1 is passed on as value for getRand, the function always returns 0.

```

170 uint64 GetRand(uint64 nMax)
171 {
172     if (nMax == 0)
173         return 0;
174
175     // The range of the random source must be a multiple of the modulus
176     // to give every possible output value an equal possibility
177     uint64 nRange = (std::numeric_limits<uint64>::max() / nMax) * nMax;
178     uint64 nRand = 0;
179     do
180         RAND_bytes((unsigned char*)&nRand, sizeof(nRand));
181     while (nRand >= nRange);
182     return (nRand % nMax);
183 }
184
185 int64 GetRandRange(int64 nMin, int64 nMax)
186 {
187     if (nMax - nMin <= 0)
188         return 0;
189
190     return GetRand(nMax - nMin) + nMin;
191 }

```

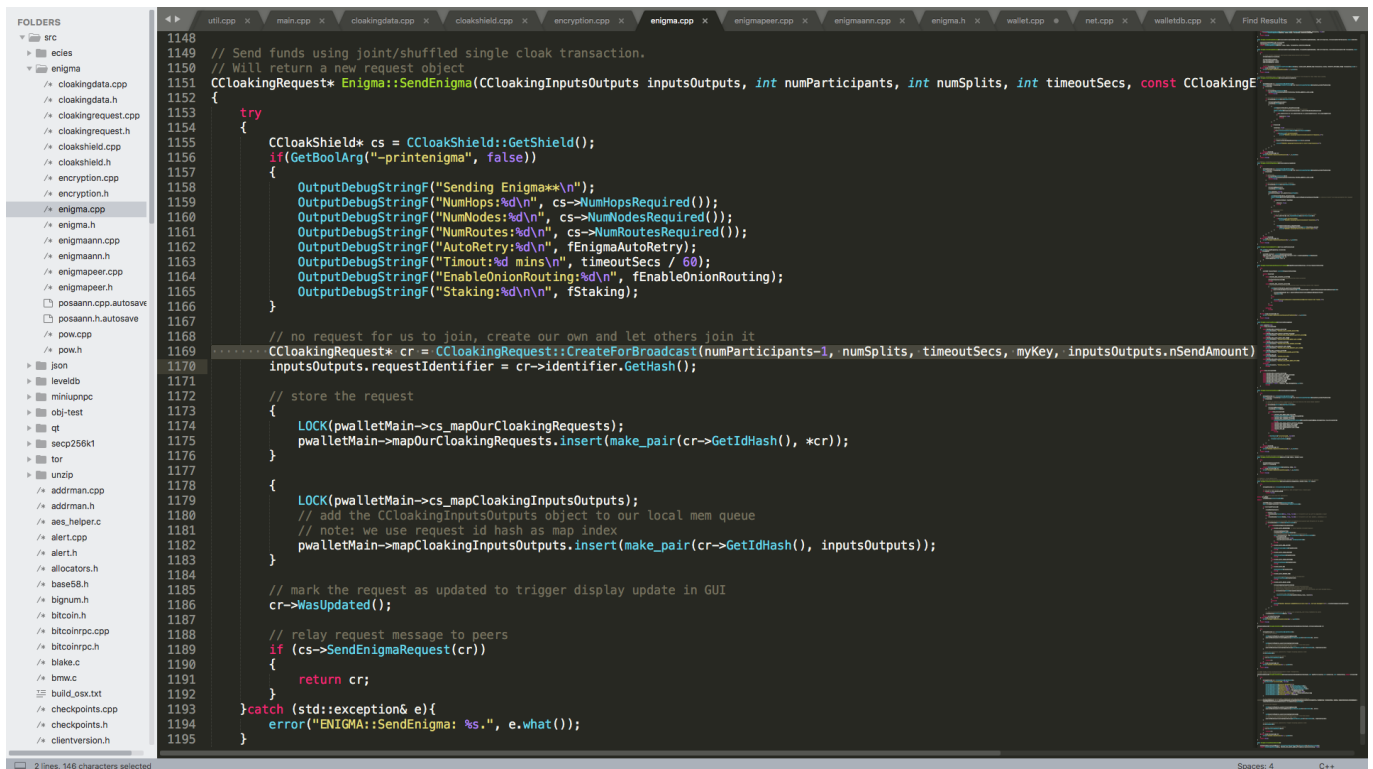
Original Bitcoin code

Added by Cloakcoin

GetRand (1) = 0, GetRandRange = nMin = 2

Figure 9.12: If the getRand function returns 0, the getRandRange function will always return 2.

9.1.8 Evidence for Vulnerability ID 11: Incorrect Number of Cloakers used



```

1148 // Send funds using joint/shuffled single cloak transaction.
1149 // Will return a new request object
1150 CCloakingRequest* Enigma::SendEnigma(CCloakingInputsOutputs inputsOutputs, int numParticipants, int numSplits, int timeoutSecs, const CCloakingE
1151 {
1152     try
1153     {
1154         CCloakShield* cs = CCloakShield::GetShield();
1155         if(GetBoolArg("--prntenigma", false))
1156         {
1157             OutputDebugStringF("Sending Enigma**\n");
1158             OutputDebugStringF("NumHops:%d\n", cs->NumHopsRequired());
1159             OutputDebugStringF("NumNodes:%d\n", cs->NumNodesRequired());
1160             OutputDebugStringF("NumRoutes:%d\n", cs->NumRoutesRequired());
1161             OutputDebugStringF("AutoRetry:%d\n", fEnigmaAutoRetry);
1162             OutputDebugStringF("Timeout:%d mins\n", timeoutSecs / 60);
1163             OutputDebugStringF("EnableOnionRouting:%d\n", fEnableOnionRouting);
1164             OutputDebugStringF("Staking:%d\n\n", fStaking);
1165         }
1166         // no request for us to join, create our own and let others join it
1167         CCloakingRequest* cr = CCloakingRequest::CreateForBroadcast(numParticipants-1, numSplits, timeoutSecs, myKey, inputsOutputs.nSendAmount)
1168         inputsOutputs.requestIdentifier = cr->identifier.GetHash();
1169         // store the request
1170         {
1171             LOCK(pwalletMain->cs_mapOurCloakingRequests);
1172             pwalletMain->mapOurCloakingRequests.insert(make_pair(cr->GetIdHash(), *cr));
1173         }
1174         {
1175             LOCK(pwalletMain->cs_mapCloakingInputsOutputs);
1176             // add the CCloakingInputsOutputs object to our local mem queue
1177             // note: we use request id hash as map index
1178             pwalletMain->mapCloakingInputsOutputs.insert(make_pair(cr->GetIdHash(), inputsOutputs));
1179         }
1180         // mark the request as updated to trigger display update in GUI
1181         cr->WasUpdated();
1182         // relay request message to peers
1183         if (cs->SendEnigmaRequest(cr))
1184         {
1185             return cr;
1186         }
1187     } catch (std::exception& e) {
1188         error("ENIGMA:SendEnigma: %s.", e.what());
1189     }
1190 }
1191

```

Figure 9.13: This caption shows the line of code where the number of participants (cloakers) is decreased by one.

sample (Read-Only)									
Home Insert Page Layout Formulas Data Review View									
Calibri (Body) 12 A A Wrap Text Number Conditional Formatting Format as Table Cell Styles Insert Delete Format AutoSum Fill Sort & Filter									
fx =E1-0,09+10									
	A	B	C	D	E	F	G	H	I
1	Address	Amount	Identified Type						
2	mt1g8.9PEfMAuuegm7GTGdhwDHI5dXSEN	0,796567	Output of cloaker or recipient		0,049757	9,959757	9,969757		
3	mkNgRAGD1wEtK8QVjNgzvOH9m4ohH4a9Kj	90,153015	Sender credit		0,111599	10,021599	10,051599		
4	mfiBlyUEs2gNEJ6NztR4bQ96JMH8dNha	395,626985	Sender credit		0,067248	9,977248	10,007248		
5	mr33vFwzWnVLGSVjG9jnfMCvCwbX3SgfUE	6,767239	Output of cloaker or recipient		0,161356	10,071356	10,101356		
6	mrLH4ET6pxQdloVRG8SgWPWMSKNnxkoaDr	0,575909	Output of cloaker or recipient		0,117005	10,027005	10,057005		
7	mjPMo4A7CvDLoxohNhlLSMDy6XYW5jff	0,049757	Split of change + reward		0,178847	10,088847	10,118847		
8	mwog94chutVZwru88Zm151Ppc1Y5d5wde	9,429189	Output of cloaker or recipient						
9	mHm73gAPGLVaJmwpfQwLevG1V1YMaB8	9,203433	Output of cloaker or recipient						
10	mnbDxMZbZagmbbfG9a8nSwLHPUsjnEnW	0,111599	Split of change + reward						
11	mxaYB8dTNCT7VBhLYag6nK7NvaMg7N3vvh	0,067248	Split of change + reward						
12	mkksfA5q2km97AsbwnQoY33TPe5KZVVCUj	9,717783	Output of cloaker or recipient						
13									
14		516,008604							
15									
16	Actual Transaction Amount:	10,000000							
17									
18									
19			Sum Yellow:	10,000000					
20			Sum Red:	10,000000					
21			Sum Green:	10,000000					
22			Credit for Sender, similar to Change + Reward	485,780000					
23	Issues:		Sum Change + Reward	0,228604					
24	Only two Cloakers (three chosen)		Sum Change without Reward	0,048604					
25	Transferred Sum can be determined (as multiple times the Outputs can be summed up to 10)								
26									
27									
28									
29									
30									
31									
32									
33									
34									

Figure 9.14: In this Excel, a transaction was analysed where three cloakers were selected. The actual receiver is marked in red, the cloakers are marked in yellow and green. Therefore, only two cloakers were actually used to cloak this transaction.