Noname manuscript No.
(will be inserted by the editor)

ZombieCoin 2.0: Managing Next-Generation Botnets using Bitcoin

Syed Taha Ali · Patrick McCorry · Peter Hyun-Jeen Lee · Feng Hao

Received: date / Accepted: date

Abstract Botnets are the preeminent source of online crime and arguably one of the greatest threats to the Internet infrastructure. In this paper, we present ZombieCoin, a botnet command-and-control (C&C) mechanism that leverages the Bitcoin network. ZombieCoin offers considerable advantages over existing C&C techniques, most notably the fact that Bitcoin is designed to resist the very same takedown campaigns and regulatory processes that are the most often-used methods to combat botnets today. Furthermore, we describe how the Bitcoin network enables novel C&C techniques, which dramatically expand the scope of this threat, including the possibilities of flexible rendezvous scheduling, efficient botnet partitioning, and fine-grained control over bots.

We validate our claims by implementing ZombieCoin bots which we then deploy and successfully control over the Bitcoin network. Our findings lead us to believe that Bitcoin-based C&C mechanisms are a highly desirable option botmasters will pursue in the near future. We hope our study provides a useful first step towards devising effective countermeasures for this threat.

Keywords botnets · Bitcoin · cryptocurrencies · C&C

1 Introduction

Botnets are networks of compromised machines, individually referred to as bots or *zombies*, controlled remotely by a malicious entity known as the botmaster. They were originally developed as tools for vandalism and to showcase hacking skills, and have evolved into sophisticated platforms geared towards financial gain and cyberwarfare. Almost eight years have passed since Vint Cerf's dire warning of a botnet "pandemic" [1], and since then the threat has only intensified.

Large botnets today typically number millions of infected victims, employed in a wide range of illicit activity including spam and phishing campaigns, spying, information theft and extortion [2]. The FBI recently estimated that 500 million computers are infected annually, incurring global losses of approximately $110 billion [3]. Botnets have now started conscripting mobile phones [4] and smart devices, such as refrigerators and surveillance cameras to spam and mine cryptocurrencies [5]. There are even national security implications: in the Estonian cyberattacks of 2007, botnets mounted distributed denial of service (DDoS) campaigns, crippling Estonian ICT infrastructure and forcing government portals, media outlets, banks, and telcos to disconnect from the Internet [6]. These alarming developments have prompted US lawmakers to actively pursue legislation to combat the botnet threat [7].

The fatal weak point for botnets is the C&C infrastructure which essentially functions as the central nervous system of the botnet. Downstream communication

This work is supported by the European Research Council (ERC) Starting Grant (No. 306994).

Syed Taha Ali
School of Electrical Engineering and Computer Science
National University of Sciences and Technology, Pakistan
Tel.: +123-45-678910
Fax: +123-45-678910
E-mail: taha.ali@seecs.edu.pk

Patrick McCorry and Feng Hao
School of Computing Science, Newcastle University,
Newcastle upon Tyne, UK,
E-mail: {patrick.mccorry,feng.hao}@ncl.ac.uk

Peter Hyun-Jeen Lee
Paysafe Group,
Cambridge, UK,
E-mail: peter.hyunjeen.lee@gmail.com

comprises instructions and software updates sent by the botmaster, whereas upstream communication from bots includes *loot*, such as financial data, login credentials, etc. Security researchers usually reverse engineer a bot, infiltrate the C&C network, trace the botmaster and disrupt the botnet. The overwhelming majority of successful takedown operations to date have relied heavily on exploiting or subverting botnet C&C infrastructures [2].

In this paper, we argue that Bitcoin is an ideal C&C dissemination mechanism for botnets. Bitcoin is a fully functional decentralized cryptocurrency, the popularity of which has skyrocketed in the wake of the global financial crisis. 1 bitcoin (or BTC) trades at approximately $480 and the currency has a market cap of approximately $6.17 billion [8].[1] Bitcoin trades over $257 million a day, which is a greater volume of transactions than Western Union and this year some expect it to overtake Paypal and American credit card networks including Discover [9]. At the heart of Bitcoin's success is the blockchain, a massively distributed, cryptographically verifiable database, maintained over the Bitcoin P2P network, which tracks currency ownership in near real-time.

Bitcoin offers botmasters considerable advantages over existing C&C techniques such as IRC chatrooms, HTTP rendezvous points, or P2P networks. First, by piggybacking communications onto the Bitcoin network, the botmaster is spared the costly and hazardous process of maintaining a custom C&C network. Second, Bitcoin provides some degree of anonymity which may be enhanced using conventional mechanisms like VPNs or Tor. Third, Bitcoin has built-in mechanisms to harmonize global state, eliminating the need for bot-to-bot communication. Capture of one bot therefore does not expose others, and an observer cannot enumerate the size of the botnet.

Most importantly, C&C communications over the Bitcoin network cannot be shut down simply by confiscating a few servers or poisoning routing tables. The Bitcoin network is designed to withstand these very kind of attacks. Furthermore, disrupting C&C communication would be difficult to do without seriously impacting legitimate Bitcoin users and may *break* Bitcoin. Any form of regulation would be a fragrant violation of the libertarian ideology Bitcoin is built upon [10]. It would also entail significant protocol modification on the majority of Bitcoin clients scattered all over the world.

We explore in detail the possibility of running a botnet over Bitcoin. Our specific contributions are:

1. We present ZombieCoin, a mechanism enabling botmasters to communicate with bots over the Bitcoin network by embedding C&C communications in Bitcoin transactions.
2. We describe how the Bitcoin paradigm enables novel C&C possibilities including dynamic upstream channels, fine-grained control over bots, and efficient partitioning of the botnet.
3. We prototype and deploy ZombieCoin over the Bitcoin network. Experimental results indicate that bot response time is generally in the range of 5-12 seconds.
4. We suggest possible countermeasures against such botnets.

We have also chosen to make the ZombieCoin source code available for purposes of academic research strictly.[2] Our goal, of course, is not to empower criminal operations, but to evaluate this threat so that preemptive solutions may be devised. This is in the spirit of existing research efforts exploring emergent threats (such as cryptovirology [11] and the FORWARD initiative [12]).

The rest of this paper is organized as follows: Section 2 presents essential background information on botnets and Bitcoin and motivates the rest of this paper. Section 3 describes the ZombieCoin protocol in detail and proposes enhancements for additional functionality. Section 4 presents our prototype implementation and experimental results. We discuss possible countermeasures in Section 5, related work in Section 6, and conclude in Section 7.

2 Background

We summarize here the evolutionary path of C&C mechanisms, followed by a brief overview of Bitcoin.

2.1 Botnet C&C Mechanisms

First generation botnets, such as Agobot, SDBot, and SpyBot (observed in 2002-2003) [13], maintain C&C communications over **Internet Relay Chat (IRC) networks**. The botmaster hardcodes IRC server and channel details into the bot executable prior to deployment, and, after infection, bots log on to the specified chatroom for instructions. This method has numerous advantages: the IRC protocol is widely used across the Internet, there are several public servers which botnets can use, and communication is in real-time. However,

[1] Bitcoin prices are prone to fluctuation. All figures quoted in this paper date to September, 2014.

[2] Interested parties are requested to contact the authors via email.

the network signature of IRC traffic is easily distinguished. More critically, this C&C architecture is centralized. Researchers can reverse-engineer bots, allowing them to eavesdrop in C&C chatrooms, identify the bots and track the botmaster. Researchers also regularly coordinate with law enforcement to legally take down C&C chatrooms, crippling the entire botnet in just one step. According to insider accounts, two thirds of IRC botnets are shut down in just 24 hours [14].

The next generation of botnets upgraded to **HTTP-based C&C communications**. Examples include Rustock, Zeus and Asprox (observed in 2006-2008). Bots periodically contact a webserver using HTTP messages to receive instructions and offload loot. HTTP is ubiquitous on most networks and bot communications blend in with legitimate user traffic. However, web domains can be blocked at the DNS level, C&C webservers can be located and seized and the botmaster can be traced.

To adapt, botmasters came up with two major innovations. Bots are no longer hardcoded with a web address prior to deployment, but with a **Domain Generation Algorithm (DGA)** that takes date and time as seed values to generate custom domain names at a rapid rate. The rationale is that it is very costly and time-consuming for law enforcement to seize a large number of domains whereas the botmaster has to register only one to successfully rendezvous with his bots in a given time-window. Conficker-C generated 50,000 domain names daily, distributed over 116 Top Level Domains (TLDs) which proved nearly impossible to block [15]. However, DGAs can be reverse-engineered. Security researchers hijacked the Torpig botnet for a period of ten days by registering certain domains ahead of the botmasters [16].

The second innovation is **Domain Flux**: botmasters now link several hundreds of destination IP addresses with a single fully qualified domain name in a DNS record (e.g. www.domain.com). These IP addresses are swapped at high frequency (as often as every 3 minutes), so that different parties connecting to the same domain within minutes of each other are redirected to different locations. Furthermore, destination IP addresses often themselves point to infected hosts which act as *proxies* for the botmaster. Yet another layer of confusion can be added into the equation by similarly concealing the Authoritative Name Servers for the domain within this constantly changing *fast flux* cloud.

The third major development in botnet C&C infrastructures is decentralized **P2P networks** which have been used by Conficker, Nugache and Storm botnets in 2006-2007. Bots maintain individual routing tables, and every bot actively participates in routing data in the network, making it difficult to identify C&C servers. However, P2P-based bots also have weak points: for instance, to bootstrap entry into the P2P network, Phatbot uses Gnutella cache servers on the Internet and Nugache bots are hardcoded with a seed list of IP addresses, both of which are centralized points of failure [17]. Security researchers have been able to detect P2P traffic signatures, successfully crawl P2P networks to enumerate the botnet, and poison bot routing tables to disrupt the botnet. In a concerted takedown effort, Symantec researchers took down the ZeroAccess botnet by flooding routing advertisements that overwhelmed bot routing tables with invalid or *sinkhole* entries, isolating bots from each other and crippling the botnet [18].

Some botnets employ multiple solutions for robustness, for example, Conficker uses HTTP-based C&C in addition to its P2P protocol [15]. More recently botnets have begun experimenting with **esoteric C&C mechanisms**, including darknets, social media and cloud services. The Flashback Trojan retrieved instructions from a Twitter account [19]. Whitewell Trojan used Facebook as a rendezvous point to redirect bots to the C&C server [20]. Trojan.IcoScript used webmail services like Yahoo Mail for C&C communications [21]. Makadocs Trojan [22] and Vernot [23] used Google Docs and Evernote respectively as proxies to the botmaster. The results have been mixed. Network administrators rarely block these services because they are ubiquitously used, and C&C traffic is therefore hard to distinguish. On the other hand, C&C channels are again centralized and companies like Twitter and Google are quick to crack down on them.

2.2 Bitcoin

Bitcoin may be visualized as a distributed database which tracks the ownership of virtual currency units (bitcoins). Bitcoins are not linked to users or accounts but to *addresses*. A Bitcoin address is simply a transformation on a public-key, whereas, the private-key is used to *spend* the bitcoins associated with that address. A *transaction* is a statement containing an input address, an output address, and the quantity to be transferred, digitally signed using the private-key associated with the input. More complex transactions may include multiple inputs and outputs. All inputs and outputs are created using scripts that define the conditions to claim the bitcoins.

Transactions are circulated over the Bitcoin network, a decentralized global P2P network. Users known as *miners* collect transactions and craft them into *blocks*,

which are chained into a *blockchain* to maintain a cryptographically verifiable ordering of transactions. Miners compete to solve a proof-of-work puzzle to insert their block into the blockchain. New blocks are generated at the rate of approximately once every ten minutes. The double spending problem of digital currencies is overcome by replicating the blockchain at the network nodes and using a consensus protocol to ensure global consistency of state.

Bitcoin was deliberately designed to resist the kind of centralization, monetary control, and oversight which restrict fiat currencies [10]. Users have some degree of anonymity[3] which may be enhanced using Tor and mixing services. The decentralized nature of the network and the proof-of-work puzzle ensures that transactions in the network cannot be easily regulated. Bitcoin can only be subverted if a malicious party in the network musters more computing power than the rest of the network combined.

Entrepreneurs and researchers have been quick to recognize Bitcoin as a new paradigm with wide application. Projects like Mastercoin [24], Colored Coins [25] and Counterparty [26] use the Bitcoin network as an underlying primitive to track 'virtual tokens' which denote financial instruments such as bonds and stocks, corporate currencies such as coupons and tickets, and even digital properties like subscription services or software licenses.

Namecoin [27], the first official fork of Bitcoin, enables users to register domains in the Namecoin blockchain as an alternative DNS outside of ICANN jurisdiction. Applications towards timestamping have also evolved: Commitcoin [28] is a research effort that embeds 'commitments' to data in the blockchain, effectively timestamping it. Similarly, Monegraph provides a proof-of-ownership service for digital artworks [29]. The OneName service [30] allows users to publicly link their names and Bitcoin addresses by inserting the corresponding details in the Namecoin blockchain.

3 ZombieCoin

Our work is the first to leverage the Bitcoin network to enable C&C communications for botnets. As we will demonstrate in the course of this paper, this new facility offers botmasters significant advantages over traditional C&C channels. Here we briefly outline the operation of ZombieCoin:

1) The botmaster generates a set of Bitcoin credentials, i.e. a key pair (sk, pk). The public-key, pk, is hardcoded into the bot binary file prior to deployment, so that bots can authenticate communication from the botmaster. Bots are also equipped with an instruction set to decode commands send by the botmaster. Our implementation, described in Section 4, consists of simple instructions such as REGISTER, PING, UPDATE, etc. with associated parameters.

2) The botnet is then released into the wild. We assume there is an infection mechanism to propagate the botnet. One common example nowadays is for botmasters to embed advertisements on webpages frequented by intended victims. When a viewer clicks on the link, he is redirected to a website hosting malicious code which executes in the background and infects his machine without his knowledge.

Upon infection, each bot generates a unique bot identifier. This may be done in various ways. For instance, Torpig bots derive an 8 byte identifier (nid) by hashing the victim's hard disk volume and serial number information [16]. Unique identifiers enable the botmaster to enumerate the botnet, and, as we will demonstrate later, exercise dynamic fine-grained control over the bots.

3) Bots then individually connect to the Bitcoin network and receive and propagate incoming Bitcoin transactions. All network communication for the botnet then proceeds as per the standard Bitcoin protocol specification described in [31]. By adhering to the standard protocol, the network behavior of the bots to an outside observer is indistinguishable from the traffic of a genuine Bitcoin user.

4) The botmaster periodically issues C&C instructions by obfuscating and embedding them into transactions. Bots identify these transactions by scanning the *ScriptSig* field in the transaction input which contains the botmaster's public-key, pk, and the digital signature (computed over the transaction) using private-key sk. Bots verify the signature, decode the instructions and execute them accordingly. These instructions may include commands to not only spy on the victim and steal his personal information, but also to undertake external attacks, such as send spam emails and launch DoS attacks on specified targets.

Next we detail various strategies to embed C&C commands in transactions.

3.1 Inserting C&C Instructions in Transactions

The most straightforward method is to insert C&C data in the **OP_RETURN output script function**. The OP_RETURN function is a recent feature included in the 0.11.0 release of the Bitcoin Core client and allows

[3] Bitcoin technically provides *pseudonymity*, a weaker form of anonymity, in that Bitcoin addresses are not tied to identity and it is trivial to generate new addresses.

users to insert up to 80 bytes of data in transactions. However, a transaction may only have one OP_RETURN script.

This inclusion is due to immense lobbying by the Bitcoin community [32]. Developers anticipate the usage of this function to be along the lines of meaningful transaction identifiers (similar to text fields in online banking transactions), hash digests of some data such as contracts [33], cryptotokens, or even index values to link to other data stores. Analysis of a recent 80-block portion of the blockchain reveals that the OP_RETURN field was used in about a quarter of transactions in that portion, indicating that this feature is proving popular [34]. One company has already launched timestamping services which rely on embedding hash data in this field [34].

This bandwidth is more than sufficient to embed most botnet commands which are typically instruction sets in the format $< command >< parameter > ... < parameter >$. For instance, the DDoS attack library for Agobot [13] contains commands: $ddos.synflood < host >< time >< delay >< port >$ and $ddos.httpflood < url >< number >< referrer >< delay >< recursive >$, etc. Agobot has over ninety such commands and they can be encoded numerically using efficient schemes like Huffman coding to fit within the 80 byte limit.

A second approach offering greater bandwidth possibilities is to embed C&C instructions as **unspendable outputs**. Prior to release of the OP_RETURN function, this was the common method by which users inserted custom data into transactions, and is used by Counterparty [26] and Mastercoin [24]. We dissect a typical Mastercoin transaction in Fig. 1. The first output address, *1EXoDusjG...*, referred to as the Mastercoin Exodus Address, identifies this as a Mastercoin transaction. The last output address is an unspendable output, which decodes into a Mastercoin transaction. Very small bitcoin values are generally associated with such outputs because they cannot be redeemed. Up to 20 bytes of data may be inserted into an unspendable output, and a single transaction may have multiple such outputs. Proof of Existence [35], a Bitcoin-based notary public service, timestamps data by inserting hash digests as multiple unspendable outputs in transactions.

Incidentally, however, Mastercoin, Counterparty, and Proof of Existence have expressed intent to switch to the OP_RETURN function [32]. As we noted, unspendable outputs are inherently wasteful. This method is also clumsy: Bitcoin clients maintain a live inventory of unspent transaction outputs ($UTXO$) to efficiently verify validity of new transactions. Clients cannot identify malformed outputs, with the result that these addresses populate the $UTXO$ data set indefinitely (since they are never spent), affecting the efficiency of the network as a whole.

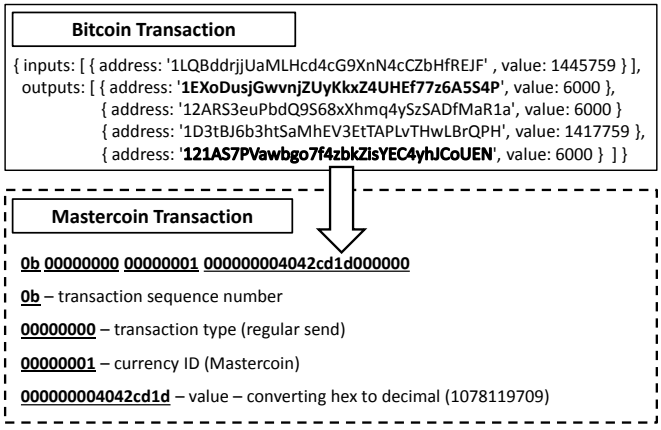

Fig. 1 Decoding a Mastercoin transaction [36]

A more elegant technique is to communicate C&C messages by **key leakage**. Signing two different messages using the same random factor in the ECDSA signature algorithm allows an observer to derive the signer's private-key d. Such instances have already been observed in the blockchain, resulting in coin theft [37]. In this case, the botmaster frames the C&C instruction within a 32 byte ECDSA private-key (including padding with random data so that identical commands do not always yield the same private-key). This is followed by an obfuscation technique to give the data enough randomness to function as a private-key. The public-key is then derived.

The botmaster then signs two transactions using the same random factor k, which will derive two signatures (r, s_1) and (r, s_2). Clearly any observer (including our bots) can detect this C&C message as r appears twice which also allows them to derive the random factor's private key, k (as outlined in [38]). Once k is known, it is then a trivial operation to derive the private signing key d and allow the bot to read the command. Notably this approach has also been used by Commitcoin [28] to insert hash digests in transactions. Bitcoins need not be wasted using this method (if the botmaster fully spends the bitcoins linked to the private key), and bandwidth is up to 32 bytes per input. However, two transactions are needed to transmit the C&C instructions.

A more covert solution is to use **subliminal channels**. Simmons [39] [40] notably demonstrated that two parties can set up a secret communications channel in digital signature schemes. This is again done by exploiting the random factor used by the signing algorithm. The botmaster creates a C&C instruction bitstring of length x bits. He then repeatedly generates signatures on the transaction using different random factors, un-

til he gets a match, i.e. a signature, the first x bits of which match the target bitstring. He attaches this signature to the transaction and publishes it. Nodes receive the transaction, verify that the signature is valid, and propagate it. Bots, on the other hand, extract the instructions from the first x bits and execute them.

Bandwidth is very restricted using this technique due to the one-way nature of the signing function. Generating x bits of an ECDSA signature to match a bitstring takes on average 2^x iterations. For larger instructions, the botmaster may choose to split the instruction into smaller target bitstrings inserted in multiple signatures. We briefly investigate here the practicality of this approach. We use an Intel i7 machine operating at 2.8 GHz with 8GB RAM, running 64-bit Windows 7, and we use the OpenSSL toolkit to construct ECDSA signatures with subliminal channels of incrementing size. In each run we construct eight signatures matching a target string and record the time taken. Results are plotted in Fig. 2.

As demonstrated, it takes under 10 minutes (600s) to *sequentially* generate eight signatures with subliminal channels of size 14 bits each. Total bandwidth in this case is $8 \cdot 14$ bits (14 bytes). We consider here a couple of optimizations: first, we use *multithreading* to parallelize operations across the multiple processors of the machine. It now takes about 3 minutes to generate eight signatures with 14-bit channels, a reduction of nearly 65%.

Second, instead of passing each thread a single target bitstring, we let each thread search across the whole range of the target bitstrings. The process stops as soon as each individual thread has located at least one distinct target. This *shared-search* step exploits the randomness of the signature generation process, increasing the odds of a successful match. We note an approximate 20% improvement over the basic multithreading scenario. It now takes approximately only 2 minutes to generate eight 14-bit subliminal channels, which is very practical. The botmaster can order the resulting signatures accordingly in the transaction to construct the complete subliminal channel.

We have considered here four methods to insert C&C instructions into the blockchain, i.e. in the OP_RETURN function, as unspendable outputs, via key leakage, and by creating subliminal channels. The botmaster can pick the technique of his choice or even combine different methods as per his requirements. While these channels are sufficient for typical botnet communications, they are however restricted in that they provide low bandwidth of only a few tens of bytes per transaction in the downstream direction (i.e. from botmaster to bots) only. However, occasionally the botmaster's communication requirements may exceed these limitations. We discuss next some novel proposals to expand the C&C communication channel.

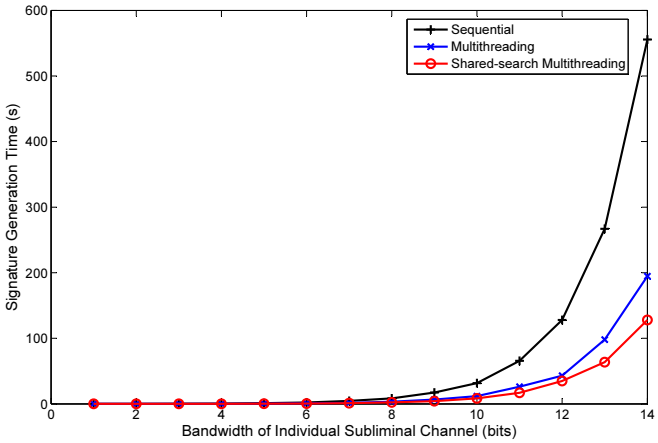

Fig. 2 Bandwidth vs. signature generation time for subliminal channels

3.2 Extending ZombieCoin

In this section, we describe enhancements to ZombieCoin to enable upstream C&C communication, delivery of larger payloads, and efficient fine-grained botnet partitioning.

Upstream Communication: Botnets require an upstream channel to send status updates and loot back to the botmaster. On successful infection, the bot usually sends a registration message (including bot identifier, machine specifications and geolocation data, etc.) and periodic heartbeat messages. Loot consists of victim's login credentials, financial data or proprietary information. It would be prohibitively expensive and impractical for bots to communicate upstream by embedding information in Bitcoin transactions. However, the botmaster may use the downstream channel to periodically announce *rendezvous points* where bots can direct upstream communications. For instance, this could be the web address of a domain owned by the botmaster.

Similar approaches have been observed in the wild. For instance, botmasters used a Facebook Wall feed to redirect Whitewell Trojan bots to C&C servers [20]. This is similar to using a domain generation algorithm but with one key difference: DGAs have been reverse engineered by researchers to lockdown rendezvous points ahead of time. Some botnets adapted by seeding DGAs with unpredictable input (such as current Twitter search trends [16]), which improves the situation a bit, but the botmaster still has to act within a very narrow time window to register domains.

In our scenario though, since the Bitcoin network acts as a near real-time broadcast channel to the bots, the botmaster can announce rendezvous points as often as he wants, and bots can start sending upstream messages right away. Typically the botmaster has a load-balancing solution deployed on servers at his end to cope with the large amount of incoming bot traffic (or it would amount to a virtual self-DDoS). To better cope, he could provide bots with multiple web addresses. Bots could even be programmed to fire a randomized timer before initiating communication.

The botmaster has considerable flexibility in this scenario. It will take time for law enforcement to neutralize his servers (depending on geographical location, ISP regulatory processes, etc.). This critical window, even if it is a few tens of minutes, may be sufficient. And if his server is shut down, the C&C channel over the Bitcoin network is still active, and the botmaster is free to try again by announcing new rendezvous points.

There is a further advantage: if bots encrypt the payload with the botmaster's public-key, they could upload the data to public locations where the botmaster could easily retrieve it. This may include services that host user-generated content such as blogging platforms like Tumbler or WordPress and cloud storage such as Dropbox, OneDrive and text-sharing services like Pastebin. These options offer less risk for the botmaster; he does not have to maintain his own servers or deploy load-balancing and location-masking services. Bot payload data is encrypted in case law enforcement confiscates it (however, the data may leak secondary information which may aid in enumerating the size of the botnet or the location of the bots). There is already a rich literature on building censorship-resistant communication channels on the Internet using social networks and public sites in a way that takedown is very hard [41] [42] [43].

Larger Payloads: As we noted earlier, the botmaster may insert multiple inputs and outputs in a transaction for greater bandwidth. An alternative for larger messages is **transaction chaining**. The botmaster splits the C&C instruction over several transactions where the output of one is the input of the next and so on. Bots receive the transactions, order them by examining the input and output fields to reconstruct the payload. We employ this technique in our proof of concept implementation, described in Section 4, to transmit 256 byte RSA public-keys to the bots. For large payloads (in the order of tens of kilobytes or more) such as software updates, the botmaster can announce rendezvous points where bots may download the data.

Partitioning Botnets: Botmasters commonly monetize their activities by partitioning botnets and leasing them as "botnets for hire" (a typical advertisement in underground markets cites a price of US $2000 for 2000 bots "consistently online for 40% of the time" [44]). Partitioning botnets also enables multitasking and is a good damage control strategy in case part of the network is compromised. The P2P Zeus botnet had over 200,000 bots, distributed into sub-botnets, by hardcoding bots with sub-botnet identifiers prior to deployment [45]. The Storm botnet assigned unique encryption keys to bots to distribute them into sub-botnets [46].

This simple approach to partitioning the network does not permit much flexibility. Ideally the botmaster should be able to partition botnets dynamically using parameters such as size, geographical location, machine specification, etc. In such a scenario, more powerful machines may be assigned to mining cryptocurrencies whereas machines with large disk space could be used to store loot. Machines in the same time zone could be used to coordinate DDoS attacks. Bots in countries with lax law enforcement may be used for spam. We present here an intuitive and elegant solution allowing fine-grained control over the botnet.

Upon successful infection, bots send a registration message to the botmaster, communicating their unique bot identifier and important information about the victim machine such as machine specification, operating system, organization, etc. The botmaster maintains a database of this information and can periodically direct queries at it[4]. Sample queries may be as follows: *What are the identifiers of all bots in the UK?* or *What are the identifiers of 1000 bots running Mac OS X?*. To direct an instruction to these particular bots, the botmaster inserts the returned identifiers into a Bloom filter and transmits the result along with the instruction by embedding the data in a Bitcoin transaction. ZombieCoin bots receive the filter result and use their identifiers to check if they are included in the set. If so, they execute the instructions. This step essentially converts the broadcast communication mode of the Bitcoin network to a multicast/anycast mode.

A Bloom filter is a space-efficient randomized data structure used to test for set membership [48]. The probability for a bot identifier that is not in the original set to result in a positive match is referred to as the Bloom filter's *false positive rate*, and is calculated as:

$$P_f = (1 - (1 - 1/m)^{kn})^k \approx (1 - e^{-kn/m})^k \quad (1)$$

where m is the size of the Bloom filter in bits, n is the number of members in the set, and k is the number of hash functions used. Minimizing P_f w.r.t k indicates

[4] C&C servers belonging to the Zeus botnet were discovered to maintain a similar MySQL database with a web-based administrative GUI for botmasters [47].

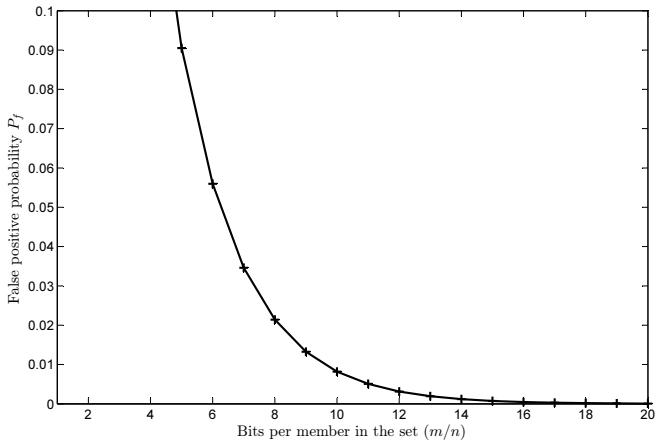

Fig. 3 False positive rate vs. number of bits per member in the Bloom filter

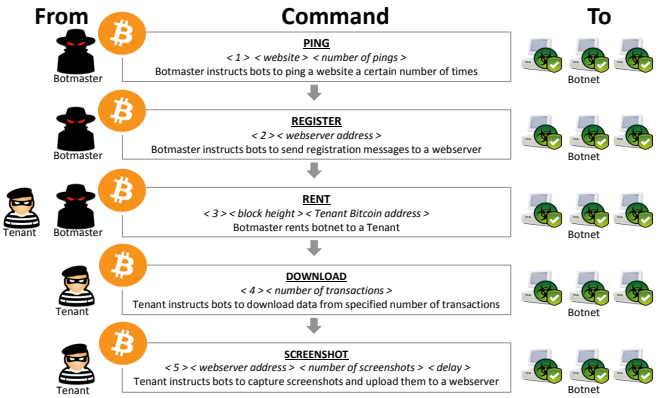

Fig. 4 Sequence of commands in the experiment

that P_f is minimum when $k = (ln2) \cdot m/n$. We plot in Fig. 3, the false positive rate for the ratio m/n, i.e. the number of bits per member.

The botmaster can now compute optimal filter parameters: to create a partition of 1000 bots with a false positive rate of less than 1% (10 bots), he will need a Bloom filter of size $10 \cdot 1000$ bits, i.e. approximately 1.2kB. For 0.5% (5 bots), this would amount to 1.5kB. The result could easily be transmitted by transaction chaining or uploading the data to a rendezvous point.

4 Proof of Concept

To validate ZombieCoin, we build a 14 node botnet and evaluate its performance over the Bitcoin network. We use the BitcoinJ library [49], which is an open source Java implementation of the Bitcoin protocol. We chose the Simplified Payment Verification (SPV) mode [50], which has a considerably low memory and traffic footprint, ideally suited for botnets. As opposed to Core nodes, SPV nodes do not replicate the entire blockchain but only a subset of block headers and filter incoming traffic to transactions of interest. Our bot application is 7MB in size and the locally stored blockchain content is maintained at 626kB. Furthermore, at the network level, the bot's traffic is indistinguishable from that of any other legitimate Bitcoin SPV client.

To simulate a distributed presence, we installed our bots in multiple locations in the United States, Europe, Brazil, and East Asia using Microsoft's Azure cloud platform [51], and ran two bots locally in our Computing Science Department. The bots individually connect to the Bitcoin network, download peer lists, and scan for transactions and by the botmaster (us).

Our experiment loops approximately once per hour through an automated cycle of rudimentary instructions in the sequence depicted in Fig. 4. We embed C&C instructions in the OP_RETURN field and in (3-bit) subliminal channels in the outputs. Bots are hardcoded with a public-key, enabling them to identify our transactions. Bots receive transactions, verify, decode, and execute them.

We simulate botnet leasing in Step 3 in Fig. 4. Botmaster and tenant sign and publish a multi-input transaction containing the RENT command. Bots verify the input signatures, record the tenant's public-key, and accept C&C instructions issued by the tenant for the duration of the lease period. The RENT transaction is a bona fide contract between botmaster and tenant and includes the lease payment in bitcoins from the tenant to botmaster.

When the tenant assumes control, he may send bots new encryption credentials or software modules. We simulate this with the DOWNLOAD command which uses transaction chaining to send bots a 256 byte RSA public-key, split over 7 back-to-back transactions. When bots receive the SCREENSHOT command, they capture a snapshot of the victim's desktop, encrypt it using the tenant's RSA public-key and send it to the web address specified.

We collect over 2300 responses from our bots over a 24 hour period.[5] We are interested in the C&C channel latency and in the time it takes for bots to respond to an instruction. We define a bot's **response time** as the time period from when the botmaster issues an

[5] The C&C transactions pertaining to our experiment can be identified in the blockchain by transaction input 1LujiuygToEddPEmRGMQUGXbsMGmup1Wrs. The initial 'ping' command is recorded in Block 319998 (transaction ID: b26b3ea0d8065d3288a5142580a5f0e372445d27bb51b45a491d2e5f20238c5e). The final 'screenshot' command occurs in Block 320153 (transaction ID: 326e06b6c187c5d97ad783fc4d7bd67cf9c80894cd9837d5e83b04ce0f0f4068). Commands can be decoded by setting the offset for each ASCII character to -125.

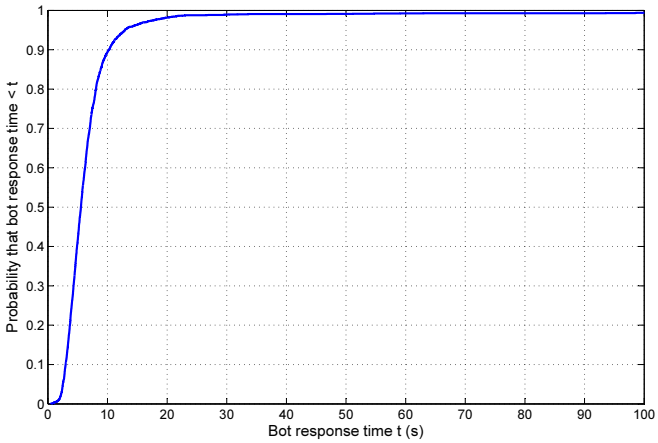

Fig. 5 Cumulative probability distribution of bot response time

instruction and it is successfully received by the bot over the Bitcoin network. To synchronize readings over multiple time zones, we configure bots to set their clocks using a common timeserver.

All bots successfully received the botmaster's instructions. Fig. 5 plots the cumulative probability distribution of the bot response time. Due to the connectivity of the Bitcoin P2P network, about 50% of the time, the bots responded within 5 seconds, and 90% of the time within 10 seconds. The median response time is 5.54 seconds. In the interest of improved visualization, our results do not show outliers beyond the 100 second mark. Only in 15 instances (0.6% of the overall communications) was bot response time greater, ranging from 100-260s.

4.1 Discussion

To summarize thus far: ZombieCoin inherits the key strengths of the Bitcoin network, namely low-latency communication, consistent network state, and a distributed decentralized architecture. The botmaster need not maintain his own C&C infrastructure, which is a risky and costly endeavor. Bots can be maintained in isolation from each other. C&C traffic over the local network is indistinguishable from that of legitimate Bitcoin users. Upstream channels can be conveniently established and Bloom filters enable fine-grained control over the botnet. We believe our experimental results, together with the relative ease of implementation using freely available software, highlight the realistic and practical aspects of ZombieCoin and we should take seriously the threat of botnets upgrading C&C communications onto the Bitcoin network.

So far we have assumed bots identify messages from the botmaster based on transaction input which raises the possibility of blacklisting the botmaster's Bitcoin address. This is not likely to resolve the problem. For one, it would be a form of regulation, a fundamental violation of the Bitcoin ethos [10], and we expect Bitcoin users would be the first to vigorously resist such attempts.

Second, such a step would require a significant protocol upgrade which could potentially degrade performance and usability of Bitcoin for legitimate users. Miners by themselves could, with relative ease, cooperate and ensure ZombieCoin transactions do not appear in the blockchain. However, this does not solve the underlying problem of the circulation of ZombieCoin transactions throughout the network. In the current Bitcoin protocol version, nodes that receive incoming transactions perform checks for correctness (i.e. the input address is valid, the transaction is in the correct format, sum of inputs equals outputs, the digital signature is verified, etc.) and then forward the transaction on to other nodes. Valid transactions are forwarded to all nodes, irrespective of the number of nodes in the network.

In our implementation described earlier, our bots do not look up transactions from incoming blocks of the blockchain (at approximate 10 minute intervals), but instead receive them within a 5-12 second window as the transactions propagate throughout the network. Therefore, even if all C&C transactions are ultimately rejected by miners, the bots have already received them, validated them, and carried out the embedded instructions. Halting the propagation of these transactions in the Bitcoin network would require the explicit cooperation of the majority of nodes in the network, necessitating not just protocol modifications, but network-wide synchronization of nodes against a blacklist that all parties agree upon.

Furthermore, to defeat any censorship measures the botmaster can switch to alternate authentication strategies which do not rely solely on Bitcoin addresses but may use subliminal channels in transaction outputs or digital signatures. Botmasters could potentially keep switching authentication strategies, thereby escalating the fight and making it harder for legitimate clients to use the network.

In theory, an anti-virus installed on a victim's machine could scan the Bitcoin network in lockstep with bots and block incoming C&C instructions. However, new malware are adept at evading anti-viruses: Torpig bots [16] contain rootkit functionality, executing their code prior to loading the OS, or injecting their code into legitimate processes to escape detection. Others like Ze-

roAccess contain tripwire mechanisms which suspend anti-virus scanning activity [18].

We would also make mention here of the costs of running ZombieCoin. At the time of our experiments, it cost us about 3 cents (0.1mBTC) for every 1000 bytes of data in the transaction. Our experiment ran over 24 hours and 250 C&C instructions were sent at a cost of US$ 7.50. We also note that since transactions are flooded to the entire Bitcoin network, the transaction fees would have remained constant regardless of the number of bots we deployed. These costs are therefore trivial compared to the profits made by successful botnets which are typically in the hundreds of thousands of dollars. Furthermore, Bitcoin-based C&C is also a considerably safer option compared to existing botnets where the odds of detection, botnet takedown, and identification of the botmaster are dramatically higher.

5 Recommendations

Thus far we have found little recognition of this threat among the Bitcoin community.[6] However, there has been some attempt made at raising awareness within the botnet and hacker communities. Interpol researchers at the BlackHat Asia conference recently demonstrated a malware which downloads specific coded strings from the Bitcoin blockchain (where they are stored as transaction outputs) and stitches them together into one command and executes it. Forbes magazine profiled this threat and others (including a preliminary version of ZombieCoin [53]), dubbing this phenomenon blockchain "pollution", and concluded on the somber note that there are as yet no easy solutions to this problem [54].

Perhaps we need to shift research focus back to traffic analysis and malware detection techniques. The new paradigm of software-defined networking (SDN) may hold some promise: there is already research suggesting SDN assists significantly in detecting malware-related anomalies at the network level [55].

We would stress here an earlier suggestion from the literature [16]: researchers and law enforcement should cultivate working relationships with registrars and ISPs to enable rapid response time to malware threats. If a botmaster announces rendezvous points over the Bitcoin network, registrars scattered over the world may need to block sites at very short notice. Incidentally, third party DNS services (such as OpenDNS, or Google Public DNS) and cloud-based security solutions (like Umbrella) may actually prove agile enough for this purpose [56].

Another approach proposed before, but, to the best of our knowledge, never applied in practice is to combat the botnet problem at its root, the economy that drives it. Ford et al. propose [57] deliberately infecting large numbers of decoy virtual machines (honeypots) to join the botnet but remain under control of the white hats. By disruptive, unpredictable behavior, these sybils will actively undermine the economic relationship between the botmaster and clients. An ad master for instance, may pay for a certain number of ad impressions, and the machines may make artificial clicks but this will not translate to a corresponding increase in actual sales. Targeting the economic incentive may prove a potent counter to the botnet threat.

6 Prior Work

Botnet-related research follows multiple strands. There are studies on the botnet economy [58] [57] [59]. Researchers have autopsied botnets, including early varieties like Agobot, SDbot [13], and state-of-the-art worms, Conficker [60], Storm [61], Waladec [62], and ZeroAccess [45]. There is extensive work on botnet tracking methods [63] [64] and traffic analysis and detection tools such as BotSniffer [65], BotMiner [66], and BotHunter [67]. Researchers have infiltrated botnets [16] and documented insider perspectives [68]. Readers interested in comprehensive surveys of the botnet phenomenon are directed to [69] [70].

There is a growing literature on exploring novel C&C mechanisms so that preemptive solutions may be devised. We summarize here a few such efforts:

Lee et al. [71] and Szabo et al. [72] propose automated botnets that derive instructions from pervasive Internet information (e.g. stock market figures or major news events). This data cannot be easily manipulated and C&C traffic blends in with legitimate user traffic. Such botnets are uncontrolled and unpredictable. This may not make economic sense, but hearkens back to earlier days when botnets were mostly built to enhance standing in the hacker community.

Starnberg et al. present Overbot [73] which uses the P2P protocol Kademlia for stealth C&C communications. The authors share our design concerns that bot traffic is covert and not easily distinguishable. However, there are critical differences: Overbot nodes carry the private key of the botmaster, and capturing one bot

[6] The Namecoin lead developer was interviewed in 2014 on the possibility of Namecoin being used to empower botnets. His response, "*Is there a real benefit for the zombie computer to use this instead of connecting to an IRC channel or else? Updatable IP? It may be less complex to get IP from hacked computers all over the world or to build a P2P botnet. As each thing that provides power to its user, it can be used in a bad or good way (as knives, secure communication software, etc).*" [52].

compromises the entire botnet's communications. Furthermore, unlike our case where instructions are circulated within seconds, for Overbot this may take up to 12 hours. ZombieCoin also requires substantially less network management as the Bitcoin network handles message routing and global consistency.

The work closest to ours is that of Nappa et al. [74] who propose a C&C channel overlaid on the Skype network. Skype is closed-source, has a large user base, is resilient to failure, enforces default encryption, and is notoriously difficult to reverse engineer, all of which are ideal qualities for C&C communications. As in our case, disrupting this botnet would significantly impact legitimate Skype users. However, unlike Bitcoin, Skype is not designed to maintain low latency global consistency of state. Furthermore, after the Microsoft takeover in 2011, Skype has switched to a centralized cloud-based architecture [75].

Researchers have also proposed novel C&C mechanisms: Stegobot [76] creates subliminal channels on social networks by steganographic manipulation of user-shared images. Zeng et al. [77] describe a mobile P2P botnet concealing C&C communication in SMS spam messages. Desimone et al. [78] suggest creating covert channels in BitTorrent protocol messages. These solutions present interesting possibilities but are not very practical, with limitations in terms of bandwidth, latency and security.

7 Conclusion

In this paper we have described ZombieCoin, a mechanism to control botnets using Bitcoin. ZombieCoin inherits key strengths of the Bitcoin network, namely it is distributed, has low latency, and it would be hard to censor C&C instructions inserted in transactions without significantly impacting legitimate Bitcoin users. ZombieCoin has a key advantage over current botnet C&C mechanisms in that common takedown techniques of confiscating suspect web domains, seizing C&C servers or poisoning P2P networks, would not be effective. Furthermore, ZombieCoin enables novel and powerful C&C communication modes, allowing botmasters to easily set up upstream channels, expand bandwidth, efficiently partition botnets, and exercise fine-grained control over individual bots. Our prototype implementation demonstrates that it is easy to implement this C&C functionality by modifying freely available software, and experimental results show that instructions propagate in near real-time on the Bitcoin network.

We believe ZombieCoin poses a credible emergent threat and we hope our work prompts further discussion and proves a step towards devising effective countermeasures.

8 Acknowledgements

This paper is an extended version of work that was first presented in February, 2015 at the 2nd Workshop on Bitcoin Research (Bitcoin15) co-located with Financial Cryptography (FC) [53].

The authors thank Hassaan Bashir, Mike Hearn, Pawel Widera, and Siamak Shahandashti for invaluable assistance with experiments and helpful comments.

References

1. Tim Weber. Criminals 'may overwhelm the web'. BBC Home, Jan. 25 2007. http://news.bbc.co.uk/1/hi/business/6298641.stm.
2. David Dittrich. So you want to take over a botnet. In *Proceedings of the 5th USENIX conference on Large-Scale Exploits and Emergent Threats*, pages 6–6. USENIX Association, 2012.
3. Alastair Stevenson. Botnets infecting 18 systems per second, warns FBI. V3.co.uk, July 16 2014. http://www.v3.co.uk/v3-uk/news/2355596/botnets-infecting-18-systems-per-second-warns-fbi.
4. Android smartphones 'used for botnet', researchers say, July 5 2012. http://www.bbc.co.uk/news/technology-18720565.
5. James Vincent. Could your fridge send you spam? Security researchers report 'internet of things' botnet. The Independent, Jan. 20 2014. http://www.independent.co.uk/life-style/gadgets-and-tech/news/could-your-fridge-send-you-spam-security-researchers-report-internet-of-things-botnet-9072033.html.
6. Joshua David. Hackers Take Down the Most Wired Country in Europe. Wired Magazine, Aug. 21 2007. http://archive.wired.com/politics/security/magazine/15-09/ff_estonia?currentPage=all.
7. Julian Hattem. Senate Dem wants to battle botnets. The Hill, July 15 2014. http://thehill.com/policy/technology/212338-senate-dem-wants-to-battle-botnets.
8. CoinMarketCap. Crypto-Currency Market Capitalizations. BitcoinTalk, Jan. 21 2016. https://coinmarketcap.com/.
9. Joseph Young. VISA: Bitcoin is no Longer a Choice Anymore. NewsBTC, Dec. 29 2015. http://www.newsbtc.com/2015/12/29/visa-bitcoin-is-no-longer-a-choice-anymore/.
10. Maria Bustillos. The Bitcoin Boom. The New Yorker, April 2013. http://www.newyorker.com/tech/elements/the-bitcoin-boom.
11. Adam Young and Moti Yung. *Malicious cryptography: Exposing cryptovirology*. John Wiley & Sons, 2004.
12. ICT-FORWARD Consortium. FORWARD: Managing Emerging Threats in ICT Infrastructures, 2007-2008. http://www.ict-forward.eu/.
13. Paul Barford and Vinod Yegneswaran. An Inside Look at Botnets. In *Malware Detection*, pages 171–191. Springer, 2007.

14. Robert Westervelt. Botnet Masters Turn to Google, Social Networks to Avoid Detection. TechTarget, Nov. 10 2009. http://searchsecurity.techtarget.com/news/1373974/Botnet-masters-turn-to-Google-social-networks-to-avoid-detection.
15. Mark Bowden. *Worm: the First Digital World War*. Atlantic Monthly Press, 2011.
16. Brett Stone-Gross, Marco Cova, Lorenzo Cavallaro, Bob Gilbert, Martin Szydlowski, Richard Kemmerer, Christopher Kruegel, and Giovanni Vigna. Your Botnet is my Botnet: Analysis of a Botnet Takeover. In *Proceedings of the 16th ACM conference on Computer and communications security (CCS)*, pages 635–647. ACM, 2009.
17. Ping Wang, Sherri Sparks, and Cliff Changchun Zou. An advanced hybrid peer-to-peer botnet. *Dependable and Secure Computing, IEEE Transactions on*, 7(2):113–127, 2010.
18. Alan Neville and Ross Gibb. Security Response: ZeroAccess Indepth. White paper, Symantec, Oct. 4 2013.
19. Brian Prince. Flashback Botnet Updated to Include Twitter as C&C. SecurityWeek, April 30 2012. http://www.securityweek.com/flashback-botnet-updated-include-twitter-cc.
20. Andrea Lelli. Trojan.Whitewell: What's your (bot) Facebook Status Today? Symantec Security Response Blog, Oct. 2009. http://www.symantec.com/connect/blogs/trojanwhitewell-what-s-your-bot-facebook-status-today [online; accessed 22-July-2014].
21. Eduard Kovacs. RAT Abuses Yahoo Mail for C&C Communications. SecurityWeek, Aug. 4 2014. http://www.securityweek.com/rat-abuses-yahoo-mail-cc-communications.
22. Takashi Katsuki. Malware Targeting Windows 8 Uses Google Docs. Symantec Official Blog, Nov. 16 2012. http://www.symantec.com/connect/blogs/malware-targeting-windows-8-uses-google-docs.
23. Sean Gallagher. Evernote: So useful, even malware loves it. Ars Technica, Mar. 27 2013. http://arstechnica.com/security/2013/03/evernote-so-useful-even-malware-loves-it/.
24. J. R. Willet. The Second Bitcoin Whitepaper, v. 0.5, January 2012. https://sites.google.com/site/2ndbtcwpaper/2ndBitcoinWhitepaper.pdf [online; accessed 22-July-2014].
25. Meni Rosenfeld. Overview of Colored Coins, December 2012. https://bitcoil.co.il/BitcoinX.pdf [online; accessed 22-July-2014].
26. Counterparty: Pioneering Peer-to-Peer Finance. https://www.counterparty.co/.
27. Ben Isgur. A Little Altcoin Sanity: Namecoin. CoinReport, July 16 2014. https://coinreport.net/little-altcoin-sanity-namecoin/.
28. Jeremy Clark and Aleksander Essex. Commitcoin: Carbon Dating Commitments with Bitcoin. In *Financial Cryptography and Data Security*, pages 390–398. Springer, 2012.
29. Daniel Cawrey. How Monegraph Uses the Block Chain to Verify Digital Assets. CoinDesk, May 15 2014. http://www.coindesk.com/monegraph-uses-block-chain-verify-digital-assets/.
30. OneName. https://onename.io/.
31. Protocol Specification. Bitcoin Wiki. https://en.bitcoin.it/wiki/Protocol_specification.
32. Richard L. Apodaca. OP_RETURN and the Future of Bitcoin. Bitzuma, July 29 2014. http://bitzuma.com/posts/op-return-and-the-future-of-bitcoin/.
33. Gavin Andresen. Core Development Update #5. Bitcoin Foundation, Oct. 24 2013. https://bitcoinfoundation.org/2013/10/core-development-update-5/.
34. Danny Bradbury. BlockSign Utilises Block Chain to Verify Signed Contracts. CoinDesk, Aug. 27 2014. http://www.coindesk.com/blocksign-utilises-block-chain-verify-signed-contracts/.
35. Jeremy Kirk. Could the Bitcoin Network be Used as an Ultrasecure Notary Service? PCWorld, May 24 2013. http://www.pcworld.com/article/2039705/could-the-bitcoin-network-be-used-as-an-ultrasecure-notary-service.html.
36. Mastercoin transaction on Bitcoin Block Explorer. https://goo.gl/dq1ra3.
37. Joppe W Bos, J Alex Halderman, Nadia Heninger, Jonathan Moore, Michael Naehrig, and Eric Wustrow. Elliptic curve cryptography in practice. *IACR Cryptology ePrint Archive*, 2013:734, 2013.
38. Don Johnson, Alfred Menezes and Scott Vanstone. The Elliptic Curve Digital Signature Algorithm (ECDSA). In *International Journal of Information Security*, 1(1). pages 36–63. Springer, 2001.
39. Gustavus J Simmons. The prisoners problem and the subliminal channel. In *Advances in Cryptology*, pages 51–67. Springer, 1984.
40. Gustavus J Simmons. The subliminal channel and digital signatures. In *Advances in Cryptology*, pages 364–378. Springer, 1985.
41. Sam Burnett, Nick Feamster, and Santosh Vempala. Chipping away at censorship firewalls with user-generated content. In *USENIX Security Symposium*, pages 463–468. Washington, DC, 2010.
42. Luca Invernizzi, Christopher Kruegel, and Giovanni Vigna. Message in a bottle: sailing past censorship. In *Proceedings of the 29th Annual Computer Security Applications Conference*, pages 39–48. ACM, 2013.
43. Tariq Elahi and Ian Goldberg. Cordon–a taxonomy of internet censorship resistance strategies. *University of Waterloo CACR*, 33, 2012.
44. Max Goncharov. Russian Underground 101, 2012. http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp-russian-underground-101.pdf.
45. Dennis Andriesse, Christian Rossow, Brett Stone-Gross, Daniel Plohmann, and Herbert Bos. Highly resilient peer-to-peer botnets are here: An analysis of gameover zeus. In *Malicious and Unwanted Software:" The Americas"(MALWARE), 2013 8th International Conference on*, pages 116–123. IEEE, 2013.
46. Ryan Naraine. Storm Worm botnet partitions for sale, Oct. 15 2007. http://www.zdnet.com/blog/security/storm-worm-botnet-partitions-for-sale/592.
47. Insight a ZeuS C&C server. http://www.abuse.ch/?p=1192, March 20 2009.
48. Burton H Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.
49. BitcoinJ: A Java implementation of a Bitcoin client-only node. https://code.google.com/p/bitcoinj/.
50. Satoshi Nakamoto. Bitcoin: A Peer-to-peer Electronic Cash System. http://www.bitcoin.org/bitcoin.pdf, 2009. [online; accessed 22-July-2014].
51. Azure: Microsoft's Cloud Platform. https://azure.microsoft.com/en-gb/.
52. interview with khalahan - namecoins lead developer, June 2014. http://coinabul.tumblr.com/post/25890690158/khalahan-and-namecoin-interview.

53. Syed Taha Ali, Patrick McCorry, Peter Hyun-Jeen Lee, and Feng Hao. ZombieCoin: Powering Next Generation Botnets with Bitcoin . In *Proceedings of the 2nd Workshop on Bitcoin Research, BITCOIN'15*, 2015.
54. Thomas Fox-Brewster. Bitcoin's Blockchain Offers Safe Haven For Malware And Child Abuse, Warns Interpol. Forbes, March 27 2015. `http://www.forbes.com/sites/thomasbrewster/2015/03/27/bitcoin-blockchain-pollution-a-criminal-opportunity/#6ae1d8583297`.
55. Syed Akbar Mehdi, Junaid Khalid, and Syed Ali Khayam. Revisiting traffic anomaly detection using software defined networking. In *Recent Advances in Intrusion Detection*, pages 161–180. Springer, 2011.
56. Chris Hoffman. 7 Reasons to Use a Third-Party DNS Service, Sept. 7 2013. `http://www.howtogeek.com/167239/7-reasons-to-use-a-third-party-dns-service/`.
57. Richard Ford and Sarah Gordon. Cent, five cent, ten cent, dollar: hitting botnets where it really hurts. In *Proceedings of the 2006 workshop on New security paradigms*, pages 3–10. ACM, 2006.
58. Jason Franklin, Adrian Perrig, Vern Paxson, and Stefan Savage. An inquiry into the nature and causes of the wealth of internet miscreants. In *ACM conference on Computer and communications security*, pages 375–388, 2007.
59. Zhen Li, Qi Liao, and Aaron Striegel. Botnet economics: uncertainty matters. In *Managing Information Risk and the Economics of Security*, pages 245–267. Springer, 2009.
60. Phillip Porras, Hassen Saïdi, and Vinod Yegneswaran. A foray into confickers logic and rendezvous points. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2009.
61. Thorsten Holz, Moritz Steiner, Frederic Dahl, Ernst Biersack, and Felix C Freiling. Measurements and Mitigation of Peer-to-Peer-based Botnets: A Case Study on Storm Worm. In *Proceedings of the First USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, pages 1–9, 2008.
62. Ben Stock, Jan Gobel, Markus Engelberth, Felix C Freiling, and Thorsten Holz. Walowdac-analysis of a peer-to-peer botnet. In *Computer Network Defense (EC2ND), 2009 European Conference on*, pages 13–20. IEEE, 2009.
63. Evan Cooke, Farnam Jahanian, and Danny McPherson. The zombie roundup: Understanding, detecting, and disrupting botnets. In *Proceedings of the USENIX SRUTI Workshop*, volume 39, page 44, 2005.
64. Daniel Ramsbrock, Xinyuan Wang, and Xuxian Jiang. A first step towards live botmaster traceback. In *Recent Advances in Intrusion Detection*, pages 59–77. Springer, 2008.
65. G Gu, J Zhang, and W Lee. Botsniffer: Detecting botnet command and control channels in network traffic. In *Proceedings of the 15th Annual Network and Distributed System Security Symposium, NDSS*, 2008.
66. Guofei Gu, Roberto Perdisci, Junjie Zhang, Wenke Lee, et al. Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection. In *USENIX Security Symposium*, pages 139–154, 2008.
67. Guofei Gu, Phillip A Porras, Vinod Yegneswaran, Martin W Fong, and Wenke Lee. Bothunter: Detecting malware infection through ids-driven dialog correlation. In *USENIX Security*, volume 7, pages 1–16, 2007.
68. Chia Yuan Cho, Juan Caballero, Chris Grier, Vern Paxson, and Dawn Song. Insights from the inside: A view of botnet management from infiltration. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2010.
69. Sheharbano Khattak, N Ramay, K Khan, A Syed, and S Khayam. A Taxonomy of Botnet Behavior, Detection, and Defense. *IEEE Communications Surveys & Tutorials*, 16(2):898–924, 2014.
70. Sérgio SC Silva, Rodrigo MP Silva, Raquel CG Pinto, and Ronaldo M Salles. Botnets: A Survey. *Computer Networks*, 57(2):378–403, 2013.
71. Hui Huang Lee, Ee-Chien Chang, and Mun Choon Chan. Pervasive Random Beacon in the Internet for Covert Coordination. In *Information Hiding*, pages 53–61. Springer, 2005.
72. Joe Szabo, John Aycock, Randal Acton, and Jörg Denzinger. The tale of the weather worm. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 2097–2102. ACM, 2008.
73. Guenther Starnberger, Christopher Kruegel, and Engin Kirda. Overbot: a Botnet Protocol based on Kademlia. In *Proceedings of the 4th international Conference on Security and Privacy in Communication Networks (SecureComm)*, page 13. ACM, 2008.
74. Antonio Nappa, Aristide Fattori, Marco Balduzzi, Matteo DellAmico, and Lorenzo Cavallaro. Take a Deep Breath: a Stealthy, Resilient and Cost-effective Botnet using Skype. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 81–100. Springer, 2010.
75. Zack Whittaker. Skype ditched peer-to-peer supernodes for scalability, not surveillance. `http://www.zdnet.com/skype-ditched-peer-to-peer-supernodes-for-scalability-not-surveillance-7000017215/`, June 24 2013.
76. Shishir Nagaraja, Amir Houmansadr, Pratch Piyawongwisal, Vijit Singh, Pragya Agarwal, and Nikita Borisov. Stegobot: a Covert Social Network Botnet. In *Information Hiding*, pages 299–313. Springer, 2011.
77. Yuanyuan Zeng, Kang G Shin, and Xin Hu. Design of SMS Commanded-and-Controlled and P2P-Structured Mobile Botnets. In *Proceedings of the Fifth ACM conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, pages 137–148, 2012.
78. Joseph Desimone, Daryl Johnson, Bo Yuan, and Peter Lutz. Covert Channel in the BitTorrent Tracker Protocol. In *International Conference on Security and Management*. Rochester Institute of Technology, 2012. `http://scholarworks.rit.edu/other/300`.