ZombieCoin: Powering Next-Generation Botnets with Bitcoin

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

Newcastle University, UK
{taha.ali,patrick.mccorry,peter.lee,feng.hao}@ncl.ac.uk

Abstract. Botnets are the preeminent source of online crime and arguably the greatest threat to the Internet infrastructure. In this paper, we present ZombieCoin, a botnet command-and-control (C&C) mechanism that runs on the Bitcoin network. ZombieCoin offers considerable advantages over existing C&C techniques, most notably the fact that Bitcoin is designed to resist the very regulatory processes currently used to combat botnets. We believe this is a desirable avenue botmasters may explore in the near future and our work is intended as a first step towards devising effective countermeasures.

1 Introduction

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 (individually referred to as bots or *zombies*), 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].

The fatal weak point for botnets is the C&C infrastructure, the central nervous system of the botnet. Downstream communication 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 offers botmasters considerable advantages over existing C&C techniques such as 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 may not easily 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. Furthermore, disrupting C&C communication would be very hard 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 [6]. It would also entail significant protocol modification on the majority of Bitcoin clients scattered all over the world.

In this paper, we explore in detail the possibility of running a botnet over Bitcoin. 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 [7] and the FORWARD initiative [8]). Our specific contributions are:

1. We present ZombieCoin, a mechanism enabling botmasters to communicate with bots by embedding C&C information in Bitcoin transactions.
2. We enumerate various strategies to embed C&C information in transactions and undertake a detailed comparison.
3. We prototype and deploy ZombieCoin and issue C&C commands to bots over the Bitcoin network. Our results show that bots receive and respond in a 5-12 second window.

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) [9], 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, 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 [10].

The second 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 very 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 [11]. 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 [12].

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 with very 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 botnet C&C infrastructure, decentralized **P2P networks**, 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 very 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 [13]. 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 [14].

Some botnets employ multiple solutions for robustness, for example, Conficker uses HTTP-based C&C in addition to its P2P protocol [11]. 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 [15]. Whitewell Trojan used Facebook as a rendezvous point to redirect bots to the C&C server [16]. Trojan.IcoScript used webmail services like Yahoo Mail for C&C communications [17]. Makadocs Trojan [18] and Vernot [19] 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.

Trustlessness is fundamental to Bitcoin: Bitcoin was deliberately designed to resist the kind of centralization, monetary control, and oversight which restrict fiat currencies [6]. Users have some degree of anonymity[1] 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.

3 ZombieCoin

Here we outline briefly how ZombieCoin works:

1) We assume the botmaster owns 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. Our implementation,

[1] 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.

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.

3) Bots then individually connect to the Bitcoin network and receive and propagate incoming transactions. All network communication proceeds as per the standard Bitcoin protocol specification described in [20]. By adhering to the standard protocol, the network behavior of the botnet to an outside observer is indistinguishable from the traffic of a genuine Bitcoin user.

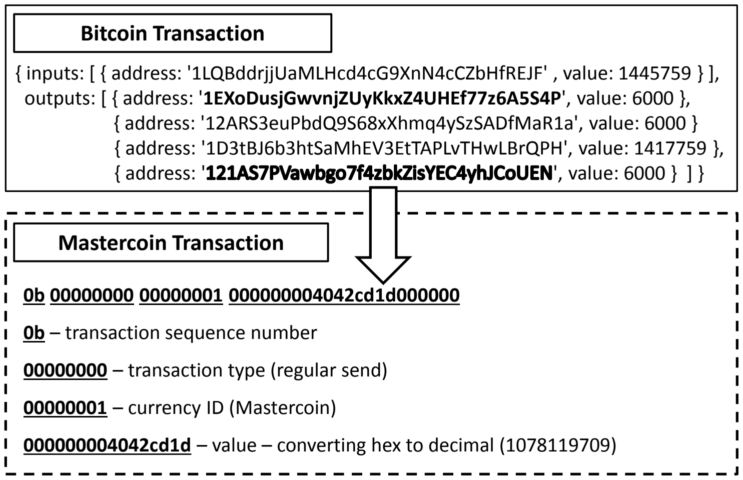
4) The botmaster periodically issues C&C instructions by obfuscating and embedding them into transactions. Bots identify these transactions by scanning the *ScriptSig* field in the input which contains the botmaster's public-key, *pk*, and the digital signature (computed over the transaction) using private-key *sk*. Bots verify the signature and decode and execute the instructions.

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**. This function is a recent feature included in the 0.9.0 release of the Bitcoin Core client, allowing users to insert up to 40 bytes of data in transactions. This inclusion is due to immense lobbying by the Bitcoin community [21]. 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 [22], 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 [23], indicating that this feature is proving popular. One company has already launched timestamping services which rely on embedding hash data in this field [23].

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 [9] 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 40 byte limit.

A second approach offering greater bandwidth possibilities is to embed C&C instructions as **unspendable outputs**. This is a common technique and used by Counterparty [24] and Mastercoin [25]. 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.



Bitcoin Transaction

```
{ inputs: [ { address: '1LQBddrjjUaMLHcd4cG9XnN4cCZbHfREJF' , value: 1445759 } ],
  outputs: [ { address: '1EXoDusjGwvnjZUyKkxZ4UHEf77z6A5S4P', value: 6000 },
             { address: '12ARS3euPbdQ9S68xXhmq4ySzSADfMaR1a', value: 6000 }
             { address: '1D3tBJ6b3htSaMhEV3EtTAPLvTHwLBrQPH', value: 1417759 },
             { address: '121AS7PVawbgo7f4zbkZisYEC4yhJCoUEN', value: 6000 }  ] }
```

Mastercoin Transaction

0b 00000000 00000001 000000004042cd1d000000

0b – transaction sequence number

00000000 – transaction type (regular send)

00000001 – currency ID (Mastercoin)

000000004042cd1d – value – converting hex to decimal (1078119709)

Fig. 1. Decoding a Mastercoin transaction [27]

Proof of Existence [26], 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 plan to migrate to using the OP_RETURN function [21]. 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.

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. Such instances have already been observed in the blockchain, resulting in coin theft [28]. 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 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 allowing bots to derive the private key.

This approach is used by Commitcoin [29] to insert hash digests in transactions. Bitcoins are not wasted using this method, and bandwidth is up to 32 bytes per input. However, two transactions are needed to leak the instructions.

A more covert solution is to use **subliminal channels**. Simmons [30] [31] 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, until he gets a match, i.e. a signature, the first x bits of which match the target bitstring. He attaches this signature

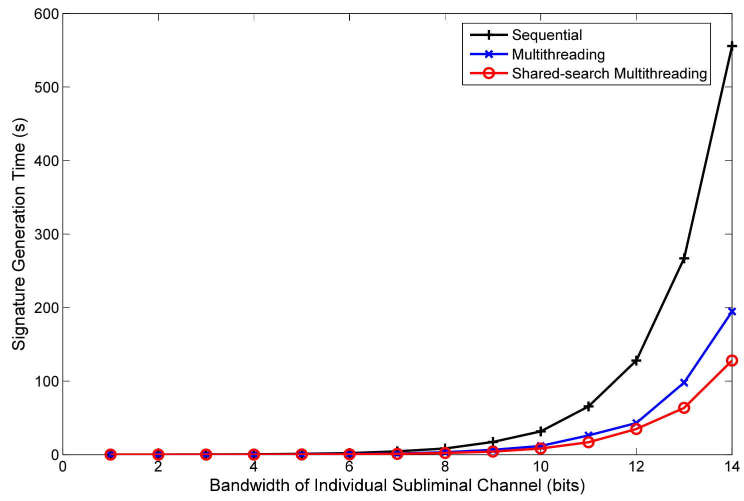


Fig. 2. Bandwidth vs. signature generation time for subliminal channels

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 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 them 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 $14 \cdot 8$ bits (i.e. 14 bytes overall). We consider here a couple of optimizations: first, we use *multithreading* to parallelize the 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, each thread now searches across the whole range of targets. The process stops as soon as each 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, taking only about 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 full channel.

4 Proof of Concept

We build a 14 node botnet and evaluate its performance over the Bitcoin network. We use the BitcoinJ library [32], which is an open source Java implementation of the Bitcoin protocol. We chose the Simplified Payment Verification (SPV) mode [33], which has a very 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, the locally stored blockchain content is maintained at 626kB, and at the network level, the bot's traffic is indistinguishable from that of any other 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 [34], 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 circulated by the botmaster (us).

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

We simulate botnet leasing in Step 3. Botmasters commonly monetize their botnet by partitioning and leasing it as "botnets for hire". In our case, botmaster and tenant sign and publish a multi-input transaction containing the LEASE command. This transaction is a bona fide contract between botmaster and tenant and includes the lease payment in bitcoins from the tenant to botmaster. Bots verify input signatures, record the tenant's public-key, and accept C&C instructions issued by the tenant for the specified lease period.

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

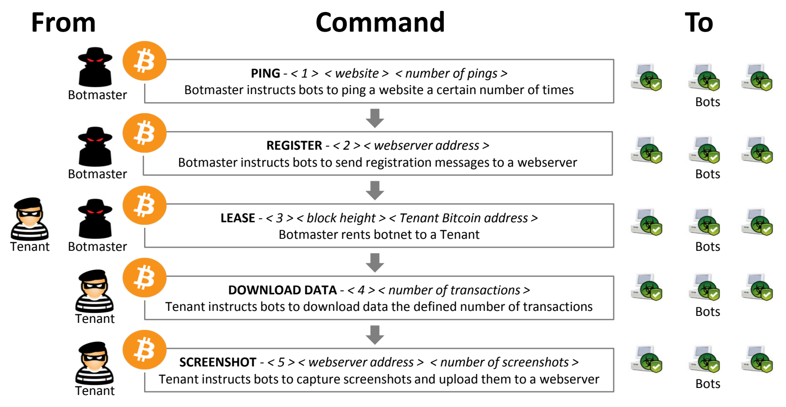

From
Command
To
Botmaster
PING - < 1 > < website > < number of pings >
Botmaster instructs bots to ping a website a certain number of times
Bots
Botmaster
REGISTER - < 2 > < webserver address >
Botmaster instructs bots to send registration messages to a webserver
Bots
Tenant
Botmaster
LEASE - < 3 > < block height > < Tenant Bitcoin address >
Botmaster rents botnet to a Tenant
Bots
Tenant
DOWNLOAD DATA - < 4 > < number of transactions >
Tenant instructs bots to download data the defined number of transactions
Bots
Tenant
SCREENSHOT - < 5 > < webserver address > < number of screenshots >
Tenant instructs bots to capture screenshots and upload them to a webserver
Bots

Fig. 3. Sequence of commands in the experiment

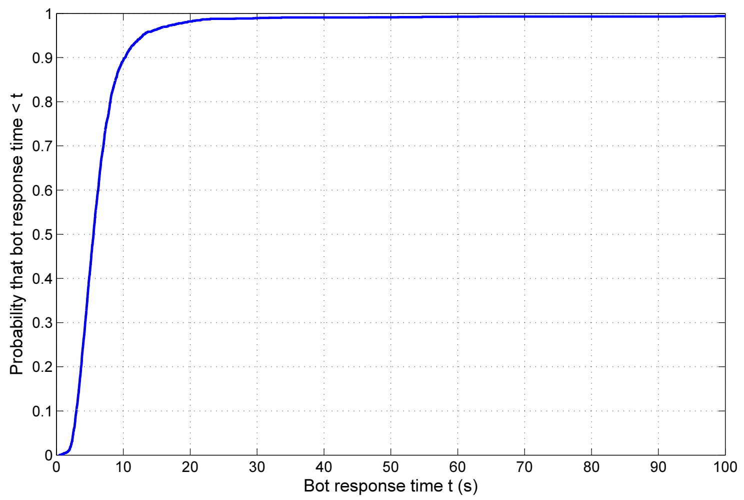


Fig. 4. Cumulative probability distribution of bot response time

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. We are interested in the C&C channel latency and in the time it takes for bots to respond to an instruction. To synchronize readings over multiple time zones, we configure bots to set their clocks using a common timeserver.

Fig. 4 plots the cumulative probability distribution of the bot response time. About 50% of the time, the bots responded within 5 seconds, and 90% of the time within 10 seconds. 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 overall results) was bot response time greater, ranging from 100-260s.

5 Discussion

We believe our preliminary results 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 [6], 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 are ignored and do not appear in the blockchain. However, this does not solve the underlying problem of the circulation of valid ZombieCoin transactions throughout the network. In the current 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, digital signature verification, etc.) and then forward the transaction on to other nodes. In our demo described earlier, our bots do not look up transactions from incoming blocks of the blockchain at approximate 10 minute intervals, but receive them within a 5-12 second window as the transactions propagate throughout the network. 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 cooperation of the majority of nodes in the network.

Furthermore, 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 escalating the fight, 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 [12] contain rootkit functionality, executing their code prior to loading the OS, or injecting their code into legitimate processes to escape detection.

We would also make brief mention here of the costs of running ZombieCoin. Typically it costs about 3 cents (0.1mBTC) for every 1000 bytes of data in the transaction. If a botmaster were to issue one command every 20 minutes, this would translate to about US$ 2.2 a day. Our experiment ran over 24 hours and 250 C&C instructions were sent at a cost of US$ 7.50. This cost is trivial not only compared to the profits of successful botnets which is typically in the hundreds of thousands of dollars, but also if one considers the alternative scenario where a botmaster runs his own customized C&C network. This dramatically increases the odds of detection, botnet takedown, and risks exposing the botmaster.

Thus far we have not found any recognition of this threat among the Bitcoin community. We urgently need constructive dialogue regarding the grave risks associated with unregulated networks. Perhaps we also 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 [35].

We would stress here an earlier suggestion from the literature [12]: researchers and law enforcement should cultivate working relationships with registrars and ISPs to enable rapid response time to malware threats. 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. [36] propose 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 botmaster and clients. An ad master for instance, may pay for a certain number of ad impressions, and sibyls making artificial clicks will not translate to the expected 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 [37] [36] [38]. Researchers have autopsied botnets, including early varieties like Agobot, SDbot [9], and state-of-the-art worms, Conficker [39], Storm [40], Waladec [41], and ZeroAccess [42]. There is extensive work on botnet tracking methods [43] [44] and traffic analysis and detection tools such as BotSniffer [45], BotMiner [46], and BotHunter [47]. Researchers have infiltrated botnets [12] and documented insider perspectives [48]. Readers interested in comprehensive surveys of the botnet phenomenon are directed to [49] [50].

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:

Starnberg et al. present Overbot [51] 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 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. [52] 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 [53].

Researchers have also proposed esoteric C&C mechanisms: Stegobot [54] creates subliminal channels on social networks by steganographic manipulation of user-shared images. Zeng et al. [55] describe a mobile P2P botnet concealing C&C communication in SMS spam messages. Desimone et al. [56] 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. Our prototype implementation demonstrates that it is easy to build 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 threat and we hope our work prompts further discussion and a step towards devising effective countermeasures.

8 Acknowledgements

This work is supported by the European Research Council (ERC) Starting Grant (No. 106591). The authors thank Hassaan Bashir, Mike Hearn, Pawel Widera, and Siamak Shahandashti for invaluable assistance with experiments and helpful comments.

References

1. T. Weber. Criminals 'may overwhelm the web'. BBC Home, Jan. 25 2007. `http://news.bbc.co.uk/1/hi/business/6298641.stm` [accessed 22-July-2014].
2. D. Dittrich. So You Want to Take Over a Botnet. In *Proceedings of the 5th USENIX conference on Large-Scale Exploits and Emergent Threats (LEET)*. USENIX Association, 2012.
3. A. 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` [accessed 22-July-2014].
4. Android Smartphones 'Used for Botnet', Researchers Say, July 5 2012. `http://www.bbc.co.uk/news/technology-18720565` [accessed 5-July-2014].
5. J. 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` [accessed 22-July-2014].
6. M. Bustillos. The Bitcoin Boom. The New Yorker, April 2013. `http://www.newyorker.com/tech/elements/the-bitcoin-boom` [accessed 22-July-2014].
7. A. Young and M. Yung. *Malicious Cryptography: Exposing Cryptovirology*. John Wiley & Sons, 2004.
8. ICT-FORWARD Consortium. FORWARD: Managing Emerging Threats in ICT Infrastructures, 2007-2008. `http://www.ict-forward.eu/` [accessed 22-July-2014].
9. P. Barford and V. Yegneswaran. An Inside Look at Botnets. In *Malware Detection*, pages 171–191. Springer, 2007.
10. Robert Westervelt. Botnet Masters Turn to Google, Social Networks to Avoid Detection. TechTarget, Nov. 10 2009.
11. M. Bowden. *Worm: the First Digital World War*. Atlantic Monthly Press, 2011.
12. B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. 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.

13. P. Wang, S. Sparks, and C.C. Zou. An Advanced Hybrid Peer-to-Peer Botnet. *Dependable and Secure Computing, IEEE Transactions on*, 7(2):113–127, 2010.
14. A. Neville and R. Gibb. Security Response: ZeroAccess Indepth. White paper, Symantec, Oct. 4 2013.
15. B. Prince. Flashback Botnet Updated to Include Twitter as C&C. SecurityWeek, April 30 2012. `http://www.securityweek.com/flashback-botnet-updated-include-twitter-cc` [accessed 22-July-2014].
16. A. 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` [accessed 22-July-2014].
17. E. Kovacs. RAT Abuses Yahoo Mail for C&C Communications. SecurityWeek, Aug. 4 2014.
18. 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` [accessed 4-Aug-2014].
19. S. 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/` [accessed 4-Aug-2014].
20. Bitcoin Wiki. Protocol Specification. `https://en.bitcoin.it/wiki/Protocol_specification` [accessed 22-July-2014].
21. R. Apodaca. OP_RETURN and the Future of Bitcoin. Bitzuma, July 29 2014.
22. G. Andresen. Core Development Update #5. Bitcoin Foundation, Oct. 24 2013.
23. D. Bradbury. BlockSign Utilises Block Chain to Verify Signed Contracts. CoinDesk, Aug. 27 2014. `http://www.coindesk.com/blocksign-utilises-block-chain-verify-signed-contracts/` [accessed 27-Aug-2014].
24. Counterparty: Pioneering Peer-to-Peer Finance. `https://www.counterparty.co/` [accessed 22-July-2014].
25. J. R. Willet. The Second Bitcoin Whitepaper, v. 0.5, January 2012. `https://sites.google.com/site/2ndbtcwpaper/2ndBitcoinWhitepaper.pdf` [accessed 22-July-2014].
26. 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` [accessed 27-Aug-2014].
27. Mastercoin transaction on Bitcoin Block Explorer. `https://blockexplorer.com/tx/17bf7d080e744df34f022dc50dc78bef8c02c2e347ba6bf4a07ceb4d20a43f86#o3`.
28. J.W Bos, J.A. Halderman, N. Heninger, J. Moore, M. Naehrig, and E. Wustrow. Elliptic Curve Cryptography in Practice. *Microsoft Research*, November 2013.
29. J. Clark and A. Essex. Commitcoin: Carbon Dating Commitments with Bitcoin. In *Financial Cryptography and Data Security*, pages 390–398. Springer, 2012.
30. G.J. Simmons. The Prisoners Problem and the Subliminal Channel. In *Advances in Cryptology*, pages 51–67. Springer, 1984.
31. G.J. Simmons. The subliminal channel and digital signatures. In *Advances in Cryptology*, pages 364–378. Springer, 1985.
32. BitcoinJ: A Java implementation of a Bitcoin client-only node. `https://code.google.com/p/bitcoinj/`.
33. S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. online, 2009. `http://www.bitcoin.org/bitcoin.pdf` [accessed 22-July-2014].
34. Azure: Microsoft's Cloud Platform. `https://azure.microsoft.com/en-gb/`.

35. S.A. Mehdi, J. Khalid, and S.A Khayam. Revisiting Traffic Anomaly Detection using Software Defined Networking. In *Recent Advances in Intrusion Detection*, pages 161–180. Springer, 2011.
36. R. Ford and S. 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.
37. J. Franklin, A. Perrig, V. Paxson, and S. 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.
38. Q Liao Z. li and A. Striegel. Botnet economics: Uncertainty matters. In *Managing Information Risk and the Economics of Security*, pages 245–267. Springer, 2009.
39. P. Porras, H. Saidi, and V. Yegneswaran. A Foray into Conficker's Logic and Rendezvous Points. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2009.
40. T. Holz, M. Steiner, F. Dahl, E. Biersack, and F. 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.
41. B. Stock, J. Gobel, M. Engelberth, F.C. Freiling, and T. Holz. Walowdac - Analysis of a Peer-to-Peer Botnet. In *Computer Network Defense (EC2ND), 2009 European Conference on*, pages 13–20. IEEE, 2009.
42. D. Andriesse, C. Rossow, B. Stone-Gross, D. Plohmann, and H. 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.
43. E. Cooke, F. Jahanian, and D. McPherson. The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets. In *Proceedings of the USENIX SRUTI Workshop*, volume 39, page 44, 2005.
44. D. Ramsbrock, X. Wang, and X. Jiang. A First Step Towards Live Botmaster Traceback. In *Recent Advances in Intrusion Detection*, pages 59–77. Springer, 2008.
45. 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.
46. 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.
47. G. Gu, P.A. Porras, V. Yegneswaran, M.W Fong, and W. Lee. BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation. In *USENIX Security*, volume 7, pages 1–16, 2007.
48. C.Y Cho, J. Caballero, C. Grier, V. Paxson, and D. Song. Insights From the Inside: A View of Botnet Management From Infiltration. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2010.
49. S. Khattak, NR Ramay, KR Khan, AA Syed, and SA Khayam. A Taxonomy of Botnet Behavior, Detection, and Defense. *IEEE Communications Surveys & Tutorials*, 16(2):898–924, 2014.
50. SSC Silva, RMP Silva, RCG Pinto, and RM Salles. Botnets: A Survey. *Computer Networks*, 57(2):378–403, 2013.
51. G. Starnberger, C. Kruegel, and E. 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.

52. A. Nappa, A. Fattori, M. Balduzzi, M. DellAmico, and L. 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.
53. Z. Whittaker. Skype Ditched Peer-to-Peer Supernodes for Scalability, not Surveillance, June 24 2013. `http://www.zdnet.com/skype-ditched-peer-to-peer-supernodes-for-scalability-not-surveillance-7000017215/` [accessed 20-July-2014.
54. S. Nagaraja, A. Houmansadr, P. Piyawongwisal, V. Singh, P. Agarwal, and N. Borisov. Stegobot: a Covert Social Network Botnet. In *Information Hiding*, pages 299–313. Springer, 2011.
55. Y. Zeng, KG. Shin, and X. 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.
56. J. Desimone, D. Johnson, B. Yuan, and P. 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`.