On the Feasibility of Automating Stock Market Manipulation

Carter Yagemann
yagemann@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia

Simon P. Chung
pchung34@mail.gatech.edu
Georgia Institute of Technology
Atlanta, Georgia

Erkam Uzun
euzun@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia

Sai Ragam
sragam3@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia

Brendan Saltaformaggio
brendan@ece.gatech.edu
Georgia Institute of Technology
Atlanta, Georgia

Wenke Lee
wenke@cc.gatech.edu
Georgia Institute of Technology
Atlanta, Georgia

ABSTRACT

This work presents the first findings on the feasibility of using botnets to automate stock market manipulation. Our analysis incorporates data gathered from SEC case files, security surveys of online brokerages, and dark web marketplace data. We address several technical challenges, including how to adapt existing techniques for automation, the cost of hijacking brokerage accounts, avoiding detection, and more. We consolidate our findings into a working proof-of-concept, man-in-the-browser malware, Bot2Stock, capable of controlling victim email and brokerage accounts to commit fraud. We evaluate our bots and protocol using agent-based market simulations, where we find that a 1.5% ratio of bots to benign traders yields a 2.8% return on investment (ROI) per attack. Given the short duration of each attack (< 1 minute), achieving this ratio is trivial, requiring only 4 bots to target stocks like IBM. 1,000 bots, cumulatively gathered over 1 year, can turn $100,000 into $1,022,000, placing Bot2Stock on par with existing botnet scams.

CCS CONCEPTS

• **Security and privacy → Economics of security and privacy**; **Distributed systems security**.

KEYWORDS

fraud, stock markets, economics, botnets, malware

ACM Reference Format:
Carter Yagemann, Simon P. Chung, Erkam Uzun, Sai Ragam, Brendan Saltaformaggio, and Wenke Lee. 2020. On the Feasibility of Automating Stock Market Manipulation. In *Annual Computer Security Applications Conference (ACSAC 2020), December 7–11, 2020, Austin, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3427228.3427241

ACSAC 2020, December 7–11, 2020, Austin, USA

ACM ISBN 978-1-4503-8858-0/20/12…$15.00
https://doi.org/10.1145/3427228.3427241

1 INTRODUCTION

Open markets are susceptible to manipulation, generating thousands of dollars of illicit profits per perpetrator per month [41]. Uncovered cases reveal manually executed scams performed by one or two coordinating people, which makes it easier to distinguish their activity from other benign traders. Most investigative breakthroughs come from uncovering links based on real life bonds (e.g., coworkers, family). This raises a serious security question: **What will happen when fraudsters start leveraging distributed automation (i.e., botnets) to conduct attacks?**

To answer this, we consider the feasibility of carrying out a market manipulation campaign using botnets. This is a radical departure from the current mindset centered around using spam [6, 17, 22, 31, 36] and social media [18, 29, 37, 50] to misinform human traders, and raises an orthogonal set of novel technical challenges, such as determining which market manipulations a hypothetical botnet would likely utilize, how bots would coordinate, whether accounts can be hijacked at the necessary scale, how they would evade detection, the impact of latency and size on effectiveness, and ultimately, how profitable this would be for the cyber criminal. By eliminating the human victim from the loop, the resulting attack achieves interesting properties, such as completely eliminating the human-readable trail left by spam and blog posts.

We present the first findings and analysis on this problem, starting with real-world U.S. Securities and Exchange Commission (SEC) case files and then gathering our own data to consider how these schemes can be fully automated by a botnet. We are the first to study the SEC's public releases from this cyber-focused perspective. From our survey of the security and default settings of online brokerages, we make the startling discovery that strong protections like two-factor authentication (2FA) are discretionary for users across the industry, making phishing straightforward. Trade notifications are by default delivered via email only, where malware with access can silently delete them before the user receives notification on any device. Dark web marketplaces sell stolen brokerage accounts containing thousands in cash for mere tens of dollars, demonstrating a low difficulty for the hackers.

From these findings, we design and implement the first proof-of-concept trading malware, Bot2Stock, and evaluate it using agent-based market simulations similar to those used to study human-driven market fraud [51] and in industry [5, 34]. Bot2Stock acts

as a man-in-the-browser, hijacking session cookies to gain access to brokerage and email accounts. Once inside, market manipulations are completed in minutes, achieving the adversary's objective before the victim can react.

From our simulations, we discover a novel and surprisingly efficient manipulation, based on layering, that does not require bots to keep each other informed about their current holdings. This dramatically simplifies the *command and control* (C2) protocol, reducing it to pointing the bots at a stock symbol and sending a global "go" signal, followed eventually by a "stop." In short, contrary to initial expectations, manipulating stocks is not a delicate tightrope act if the protocol is cleverly designed.

What we find most fascinating about Bot2Stock is unlike other malware-driven scams, our modeled criminal profits *indirectly* by provoking market movements predictable to the botmaster, but unexpected and hardly noticeable by everyone else. Distributing the orders across many accounts with no obvious correlation, in an environment where 85% of trade volume is already generated by benign bots [19], obscures the criminal's activities, hiding the paper trail that would otherwise be obvious — and subsequently "frozen" by regulators — in a "smash and grab" approach.

The results of our simulator evaluation shows that Bot2Stock is robust and profitable for the botmaster across varying botnet sizes and network latency. A 1.5% ratio of bots to background traders yields an average ROI of 2.8% for the botmaster over 2.5 seconds. Given the attack's short duration, satisfying this ratio is trivial, requiring only 4 bots per attack to target stocks like IBM. A modest network of 1,000 bots, *cumulatively* gathered over a 1-year campaign, can achieve a non-compounding annual return on investment (ROI) of 1,022% if 1 attack is performed per day over 252 trading days without reusing a bot. In other words, a botmaster can turn $100,000 into $1,022,000 in a year, which is consistent with the uncovered human-driven campaigns described in SEC reports and comparable to other botnet-based scams (e.g., click fraud). Our dark web scraper observed 1,005 stolen Charles Schwab accounts sold in 3 months on 1 marketplace, demonstrating the feasibility of gathering 1,000 bots over a yearlong campaign.

To promote future work, we have made our PoC malware and simulations publicly available.[1] This work has been disclosed to the SEC and Financial Services Information Sharing and Analysis Center (FS-ISAC).

2 RELATED WORK

Attacks & Measurements. Research on stock market manipulation can be divided into two camps. The first studies how rumors in spam [6, 17, 22, 31, 36] and social media [18, 29, 37, 50] lead to unusual market returns and volatility. Our work distinguishes itself from these by focusing entirely on trade-based manipulation, as opposed to social engineering.

The other group investigates trade-based manipulation without spreading rumors or spam [1, 2, 25, 35]. Khwaja et al. [24] investigated situations where brokers colluded and discovered that intermediaries can earn annual rates of return that are 50 to 90 percentage points higher than regular traders. This group of work most closely relates to ours, but does not consider the potential to distribute and automate manipulations to obfuscate the criminal's participation. There is recent work by Xu et al. [53] analyzing pump-and-dump in the context of crypto-currency trading, but these scams are also conducted manually by humans coordinating over services like Telegram and Discord and without layering. While there is a past work on simulating spoofing (layering), also using agent-based simulation [51], it was based on Nash equilibrium, unlike ours, and we are the first to incorporate a model for computation delay and network latency in order to simulate a C2 protocol. Prior work did not consider colluding parties (bots).

Defenses. Automatically detecting manipulation has also been explored using complex time series [20], hidden Markov chains [7, 10, 47], and other techniques [11, 20, 27, 47]. Unfortunately, accuracy remains unsatisfactory. The most accurate system we are aware of, proposed by Cao et al. [11] to detect wash trading, only achieves 45% precision and 55% recall on a dataset from the NASDAQ and London Stock Exchange.

3 BACKGROUND & OVERVIEW

3.1 Market Mechanics

Basic trading consists of creating *asks* (offers to sell a quantity of *shares* in a stock at a particular price) and *bids* (offers to buy).[2] A *limit* order will only match offers at a certain price (or better) whereas *market* orders immediately match the best available offers. These orders can be set to expire automatically after a certain time (e.g., when the market closes at the end of the day) or remain active until they are filled or explicitly canceled by the trader. Active orders are considered *open* whereas completed or canceled ones are *closed*. A trader's *positions* in a stock is the aggregate of all the shares they currently own [45].

Canceling is important to this work because it allows traders to create orders purely with the intent of canceling them later. These are referred to as *non-bona fide* orders and they form the cornerstone for some of the fraud we consider in Subsection 3.2. Making non-bona fide orders is illegal in the U.S., but since canceling is not inherently illegal, the distinction from bona fide orders is a matter of determining the trader's intent. As evident by the activities of *day traders* (professionals that watch the markets and make multiple trades daily) and high-frequency trading, even canceling high volumes of orders is not inherently illegal, which creates an opportunity for abuse.

Another important mechanism is margin trading, which consists of *margin buying* and *short selling* (*shorting*). These are analogous to loans a trader can use to borrow stock shares or cash. In short, if a trader believes a stock's price will decline, but does not currently own any shares to sell, he can borrow shares from a lender with the promise of returning them (with interest) later. This lets him sell, and then if the price does decline, he can buy back the shares he owes at the lower price, yielding a net profit. Margin buying is the same concept, except with cash. The trader borrows money from a lender, enabling him to buy more shares to reap greater gains.

[1] https://github.com/carter-yagemann/Bot2Stock

[2] Readers may already be familiar with *bidding* for items on eBay or seeing an *asking* price on Craigslist.

Table 1: Cases of Market Manipulation Selected for Further Analysis

Case Title	Fraud	Start Date	Duration	Instances	Revenue	Annual Rev.
SEC v.s. Joseph P. Willner	PnD	Sep 2014	2 Ys	110	$700,000	$350,000
SEC v.s. Unknown traders and JSC PAREX Bank	PnD	Dec 2005	1 Y	16	$732,941	$732,941
SEC v.s. Taub et al.	Layering	Jan 2014	1 Y, 11 Ms	23,000	$26,000,000	$13,565,217
SEC v.s. Milrud	Layering	Jan 2013	2 Ys	≥1	$24,000,000	$12,000,000
SEC v.s. Briargate Trading, LLC	Layering	Oct 2011	1 Y	242	$525,000	$525,000
SEC v.s. Visionary Trading LLC et al.	Layering	May 2008	2 Ys, 6 Ms	≥1	$984,398	$393,759
SEC v.s. Hold Brothers On-Line Investment Services LLC et al.	Layering	Jan 2009	1 Y, 9 Ms	325,000	$1,800,000	$1,028,571

3.2 Case Studies

From our initial study of the SEC case files dating from 2005 to 2018, Table 1 lists several exemplar instances of *pump-and-dump* (PnD) and layering frauds, including when they occurred and how long the campaigns ran before being caught. We summarize these cases here to establish a preliminary understanding of the current state of market fraud. The attacks are generalized and automated in Section 4.

Pump-and-Dump. Willner (Row 1, Table 1) prepared his scam by placing asks to short shares of the target stock at prices significantly higher than the current market value. He then used a hijacked victim account to place a matching bid at the same high price, causing his ask to execute. The victim now owns the overpriced shares. Willner then forced the hijacked account to sell the shares back to him at below market value, resulting in a significant profit for Willner and loss for the victim.

In the Unknown Traders case (Row 2, Table 1), the criminals prepared by purchasing shares in the target stock using their own accounts. Meanwhile, they liquidated existing stock shares in hijacked victim accounts into cash. They then used the resulting cash to purchase large volumes of shares in the target stock, causing a surge in the market and pumping up the price. The culprits could then sell their shares at the peak of the price surge, yielding a profit.

Layering. All the listed cases (rows 3 through 7) start with opening multiple asks to sell the target stock at progressively lower prices. A few of these orders are allowed to fill, deflating the price and applying downward *pressure* (i.e., expectation among background traders that the price will continue to decline). Some background traders sell their shares to avoid the anticipated decline, further deflating the price. The criminals respond by executing bona fide buys at the deflated price and then cancel all their remaining sell orders. With the pressure suddenly gone, the price reverts back to its original value, allowing the criminals to cash out for a profit.

The scam is then repeated in the opposite direction, starting with a series of non-bona fide buy orders at increasing prices to artificially inflate the price, followed by bona fide sells and cancels. In summary, by creating the illusion of pressure, these criminals deceive background traders into moving the stock price in their favor.

3.3 Technical Challenges

Going from manual market manipulation to automated botnet campaigns is not a one-to-one translation. There are several technical challenges an adversary has to overcome to successfully adapt the current techniques. The ones addressed in this work are:

(1) Which types of manipulation are suitable for botnet automation? (Section 4)
(2) How can a botnet bootstrap these attacks (e.g., acquire the necessary shares to begin the manipulation), and evade detection by brokerages? (Subsections 4.1, 4.2)
(3) How much trading leverage, vital for conducting fraud, does a hijacked account offer verses the difficulty of compromising it? (Subsection 4.3)
(4) How would a botnet optimize profitability (e.g., avoiding accumulating commission fees)? (Subsections 4.4–4.6)
(5) How will the botmaster compromise enough accounts, without depleting them of cash, to conduct the campaign while remaining undetected? (Subsection 5.1)
(6) How will the botmaster holistically combine these adapted techniques into a functional malware, including the hiding of notifications and transaction histories? (Subsection 5.2)
(7) How efficient can the botmaster make the C2 infrastructure while maintaining robustness and effectiveness? What ROI can he expect and how many bots does he need? (Section 6)
(8) How profitable is botnet-based stock market manipulation compared to other scams? (Subsection 7.2)
(9) What are the possible paths towards defending against a market manipulating botnet? (Subsection 7.3)

We identify these challenges as being unique to automating *market manipulation* and find that their solutions have a significant effect on the design of Bot2Stock's malware and C2 protocol. General challenges to operating botnets, like acquiring C2 infrastructure and infecting victim machines with malware, are already studied in prior and ongoing research [4, 39].

3.4 Threat Model

In this work, we assume a botmaster has assembled a botnet by infecting devices with malware and has discovered that some portion of these devices are regularly used to access brokerage accounts — presenting an opportunity to commit market fraud. We focus on U.S. exchanges and brokerages, but believe our findings are applicable to other countries as well. Users conduct trading either via their browser, using a brokerage website, or with a native program (i.e., trading platform). Accounts are protected using current industry practices, meaning 2FA is available, but not required, and notifications are delivered via email. We confirm this to be the case for the top U.S. online brokerages in Subsection 5.1.

We assume each compromised account has at least $5,000 in cash (measured in Subsection 4.3) and does not have a line of credit with the brokerage, meaning shorting and margin buying is unavailable (Subsection 3.1). The brokerages are expected to be running anomaly detection systems designed to monitor individual accounts for irregular or questionable trading patterns. For example, one complaint filed by the SEC describes Q6, a proprietary program capable of detecting indicators of layering [43]. Exchanges are also assumed to be using anomaly detection software, however due to how orders are routed from trader to brokerage to broker to exchange, the relationship between orders and traders is opaque — as evident by ongoing efforts to design and implement a consolidated audit trail for U.S. markets [13, 15]. In short, for the botnet to be successful, it must shape and distribute its trading patterns to evade several layers of anomaly detection while still maintaining profitability.

4 GENERALIZING MARKET FRAUD

The SEC publicly releases all the case files for formally investigated complaints. As a starting point for our study, we manually examine all the summaries for cases filed between 2015 and 2018 (about 700 cases in total). *We decide to focus on two types of cases, PnD and layering, because these offenses are purely based on trading behavior, making them feasible to automate.* We do not consider crimes like insider trading and Ponzi schemes, which have a significant social engineering component that a botnet alone cannot conduct.

With our categories chosen, we further expand our dataset by searching the SEC site for documents containing relevant keywords and by consulting the "Recent Trade Surveillance Enforcement Actions" list maintained by Trillium [49].

In order to identify the exemplar SEC cases for detailed analysis, we apply the following search criteria:

(1) We only consider *pure trading manipulation* and eliminate cases with non-trading behavior. For example, even if a case involved PnD, if it also relied on sending spam to promote the stock, we exclude it.
(2) We only consider cases where at least one detailed example of the alleged manipulation is provided. It must include sufficient information to derive the exact orders made and the criminal's net gain. In most cases, the culprit repeats the manipulation several times, so their total gain is significantly higher than the net gain of the lone recorded example.
(3) We only consider cases where market manipulation directly benefits the culprit. In several instances, fraud was performed to keep the stock eligible for trading on the NASDAQ, which is a motive outside the scope of this work.

A summary of all the cases we pick for further analysis is contained in Table 1. The format "x v.s. y" indicates that x is the plaintiff and y is the defendant in the case. Interestingly, among the SEC cases we consider, the majority were instances of PnD that involved non-trade behavior (mainly to promote the target). The two cases listed in Table 1 utilized hijacked accounts to *manually* carry out illegitimate trades, suggesting that this direction is ripe for malicious automation.

4.1 Automating Layering

Layering can be utilized to raise *or* lower the price of a target stock. For simplicity, we will explain how it is used to *lower* the price. The perpetrator begins by placing asks at prices slightly worse than the current best offer to create pressure. As orders from other traders execute in response to the pressure, the perpetrator cancels and replaces his own, always staying slightly worse than the best offer. Once the price is sufficiently lowered, the perpetrator bids at the manipulated price, allowing him to obtain shares at an artificially low price. He then cancels all his remaining open orders, waits for the price to revert back to its original value, and then sells his shares for a profit. Readers can refer to the Appendix for more detailed examples with real-world data.

Automation. The main challenge we notice with automating layering is a bootstrapping hurdle where *in order to make any sell orders (even non-bona fide ones), the seller must first own the shares to be sold.* It is possible to use short selling to sidestep this issue, but as previously mentioned, we do not assume the hijacked accounts have the credit necessary for margin trading. Having the bots buy shares in advance is difficult because buying too quickly will impact the price negatively for the criminal and buying too slowly will prolong the attack duration, reducing profits and raising the risk of premature detection.

Instead, the easiest solution is to start with layering *bids*, whereas the real-world cases in Table 1 all started with asks. The criminal's profit remains unchanged, but now he only needs to prepare his own account to sell as opposed to having to prepare all the bots.

Evasiveness. There are two patterns that are typically used by anomaly detectors to spot layering. The first is a trader opening orders on both sides of the order book (asks and bids), with one side being disproportionately larger than the other. In one case the SEC investigated, this kind of trading behavior tripped the anomaly detector used by Lek Securities Corporation — a proprietary system called Q6 [43]. Bot2Stock avoids creating this pattern by having each bot only place a single order, on one side of the market, to setup the layering. The other signal is placing orders on one side of a stock's order book at progressively increasing prices and then canceling them in bulk (e.g., Taub et al. [44]). Bot2Stock also avoids creating this pattern because again, each bot only places a single order. Cumulatively, an imbalance is created in the book, but individually the bots are only placing one order per account at a time.

This highlights the advantage of performing market manipulation with a botnet. Namely, the large number of bot participants obscures the manipulative intent of the botmaster by making the orders appear as the independent acts of unrelated parties.

4.2 Automating Pump-and-Dump

A PnD perpetrator starts by buying or selling shares in a stock to drive the price in the corresponding direction. Other traders see this momentum and start placing orders based on the wrong assumption that there is a legitimate reason behind the price movement. The perpetrator then stops pumping and reverses the direction of his orders to profit from the momentum. For example, if he was *buying* to pump *up* the price, he would follow up with *asks* at an even

Table 2: Stolen Charles Schwab Accounts (9/16/16–12/12/16)

Selling Price	Min Account Cash	Max Account Cash
$50	$5,000	$20,000
$75	$20,000	$100,000
$100	≥ $100,000	-
Accounts Sold	(1,005)	

Table 3: Automating Taub et al. Layering Instances

Layering Orders	Cash Lost	Cash Needed	Time (s)	Profit
Table 9 & 8	$728	$401, 102	141	$3, 285
Table 10 & 11	$1, 345	$303, 460	120	$4, 927
Table 12 & 13	$3, 065	$219, 449	218	$24, 501

higher price, yielding a profit when the momentum perpetuated by the fooled traders reaches that price.

More aggressive perpetrators can also double their profit using margin orders. Continuing the current example, when the perpetrator places his asks to dump the shares he acquired while pumping, he can also short the stock. In other words, the perpetrator borrows shares, sells them at the pumped up price and then after the price crashes back to its starting value, the perpetrator can buy at the original value to pay back the owed shares.

The primary shortcomings with PnD, from the criminal's perspective, is that acquiring enough shares or cash to bootstrap the scam is difficult and executing many trades accumulates commission fees that eat into the criminal's leverage. This leads us to conclude that layering is better suited to automation.

Automation. Table 1 contains additional details about the PnD cases, including the number of times the fraud was successfully performed and the illicit revenue according to the SEC. Applying PnD to a malware requires the botmaster to coordinate the use of the hijacked accounts. Specifically, performing the setup too slowly prolongs the attack and raises the risk of premature detection, but moving too quickly will result in a "smash and grab" with a higher risk of tripping anomaly detectors. Due to how orders are matched best-offer-first, the bots will need to keep pumping the price until it is sufficiently inflated (via buying) or deflated (via selling) for the botmaster to make a profit.

Evasiveness. In the Willner case, it took 4 days for the brokerage to start an investigation into his trading patterns. The FS-ISAC also started questioning his trades in roughly the same amount of time. The key anomaly implicating him was the placement of sell orders at prices significantly higher than the market price. The hijacked accounts were not discovered until a later investigation. Therefore, the botmaster should be careful not to place his orders prior to the bots pumping the stock's price.

Conversely, in the Unknown Traders case, the account compromises were detected first and then correlated with trades made by the culprits across 15 stock symbols. In short, reusing the exact same accounts across sessions led to their detection. They only evaded prosecution because they were able to act anonymously through the domestic brokerage accounts of Latvian-based relief defendant JSC Parex Bank. The bank was fined for negligence.

Both cases demonstrate a limitation in automating PnD. Namely, because the profiting account has to make trades that so blatantly contradict the background market movements, the botmaster is likely to be detected, regardless of the botnet's activity. This makes layering the more likely technique to be used in a successful criminal botnet campaign.

4.3 Approximating Leverage & Availability of Hijacked Accounts

Once the criminal has gained control of the victim's brokerage account, any owned assets can be used to manipulate the market. Our modeled criminal does not touch any of the securities already held in the account, because rapidly liquidating an account's assets is a red flag for brokerage anomaly detectors [42]. However, he can use the cash in the account that is readily available for trading. We refer to the combined cash across all the accounts controlled by the adversary as the *trading leverage*.

Methodology. To estimate how much leverage an adversary can expect to gain per compromised account, we turn to dark web marketplaces for data. Our data was collected over several months in 2016 from the now dismantled AlphaBay marketplace, which was accessible via the Tor network. We focus on accounts belonging to Charles Schwab, which is a major U.S. brokerage.

We collected and parsed listings using a website scraper to perform periodic keyword searches [26]. Listings on AlphaBay displayed the price and number of units (i.e., accounts) sold. Criminals priced accounts based on the amount of cash they contained. However, only the approximate cash values were displayed. We use this data as a proxy for how much effort the hacker exerted to hijack accounts. Specifically, we divide the price of the stolen account by the amount of cash it contained to approximate how much cash leverage is gained per dollar spent.

Results. Our data is summarized in Table 2. Between September and December of 2016, we found 1,005 sold accounts, priced from $50 to $100 per account. They were advertised as having at least $5,000 in cash and some claimed to contain over $100,000, although the upper bound was not provided.

Based on the data, the average leverage is $660 per every $1 spent with a minimum of $100 per $1 spent. To keep our calculations throughout this work conservative, we use the minimum observed leverage for an account that was actually sold. This stolen account went for $50 and contained $5,000 in cash.

4.4 Profitability of Automated Layering

Table 1 shows the alleged illicit gains for our layering case studies. For the rest of our analysis, we focus on the the complaint "SEC v.s. Taub et. al." because it has the most detailed transaction logs. To approximate the profitability of automated layering, we consider the revenue and costs associated with a botnet performing the same actions as reported in this case.

The full transaction tables for Taub et al. are in Appendix A. The case does not state which stocks were manipulated, but mentions that each instance was for a different company traded on the NYSE.

Table 4: Automating Pump-and-Dump Instances

Instance	Stock	Cash Lost	Cash Needed	Time (s)	Profit
Willner #1	FCCO	$2,942	$7,991	N/A	$2,942
Willner #2	HIHO	$3,600	$25,722	600	$3,000
Willner #3	EARS	$6,660	$35,460	60	$6,201
Unknown Traders #1	REDI	$49,000	$251,964	9000	$75,720
Unknown Traders #2	DEPO	$133,107	$1,435,400	3600	$51,078
Unknown Traders #3	ORCH	$77,285	$765,310	4560	$55,783

To be conservative, we assume a $5 commission fee per executed order[3] and that each hijacked account starts with at least $5,000 in cash (Subsection 4.3). To approximate the total commission fees, we take the cash value of the executed order and divide by the per-bot cash to determine the number of distinct orders bots would have to place collectively to achieve the desired result. For example, if $8,000 worth of shares were bought in the original manipulation and each bot has $5,000 in cash, 2 bots would collectively need to make 2 orders ($4,000 each), resulting in $10 of commission.

Results. Our results are summarized in Table 3. If we consider a scenario where each account costs $50 to hijack and has $5,000 in cash (Subsection 4.3), repeating the reported instances will require up to 80 bots and yield $24,501 in profit. Subtracting fees and commissions, each bot will yield $41 to $557 in profit. If we assume each bot is only used once, our lower bound profit estimate for 1,000 bots is $41,000.

Compared to PnD (calculated in Subsection 4.5), this seems like a low profit margin, especially considering that both techniques require roughly the same amount of cash to bootstrap. However, layering takes significantly less time to perform (under 4 minutes compared to hours for PnD) and inflicts almost no cash loss on the bots. Specifically, only $3 in cash is lost per bot, on average, because most non-bona fide orders cancel successfully. This makes it highly feasible to perform layering multiple times in the same span of time as a single PnD manipulation.

For example, given the cash needed to launch an instance of layering, 1,000 bots ($5 million starting cash) can conduct 12 to 22 layering manipulations in parallel. This allows 44 to 80 instances to complete in 10 minutes, generating $158,000 to $1,078,000 in profit.

4.5 Profitability of Automated Pump-and-Dump

To determine the cost and number of hijacked accounts rendered unusable after manipulation, we extract the target stock and log of fraudulent orders (identified by the SEC) from the two case files. We compute the net cost of these orders along with the total leverage needed to execute them and report the yielded profit and duration of each manipulation session.

We only focus on the orders made to manipulate the market and not the ones made by the culprit in advance to reap the benefits of the manipulation since these orders do not affect the botnet's operation.

Results. Our results are summarized in Table 4. Based on the assumption that an account holding $5,000 in cash costs $50 to hijack (Subsection 4.3), the method from Willner's case would require 2 to 8 accounts and generate at least $3,000 in profit (first three rows of Table 4).

If we consider a modest botnet of 1,000 bots (i.e., 1,000 hijacked accounts) and assume accounts are not reused across rounds, even under the worst case scenario, it could generate $750,000 in profit. These profits easily offset an upper-bound commission fee of $5 per order.

In comparison, the manipulation technique used in the Unknown Traders case yields less profits for the botmaster, but also better preserves the cash in the hijacked accounts. In the worst case, 50 to 300 hijacked accounts are needed, yielding at least $51,000 in profit (last three rows in Table 4). A botnet of 1,000 hijacked accounts would yield a lower bound of $228,000.

4.6 Botnet Self-Sustainability

Based on the numbers in Table 3, the profit for the criminal is greater than the collective cost to the bots. In other words, the criminal can feasibly use layering or PnD to achieve self-sustainability for the botnet. To achieve this, the botmaster can take turns with the bots in a round-robin fashion. When it is the botmaster's turn, he will buy stock at the deflated price and sell at the inflated price, earning a profit. When it is the bots' turn, they will collectively use their own capital to buy stock at the deflated price and sell at the peak, yielding their own profits.

To analyze the feasibility of this sustainable design, we extract from the transaction logs the capital needed for the winner for each provided example, which is at least $336,000. The results of our analysis is summarized in Table 5. As the table shows, the profit made by the bots in their round covers the loss from both the botmaster and bot rounds. Thus, under the sustainable design, not only will the bots (i.e., the hijacked accounts) never run out of cash, they will slightly gain. This surprising result distinguishes stock market manipulation from other forms of botnet scamming.

5 IMPLEMENTATION

Using our findings from Section 4, we create a fully functional proof-of-concept malware, Bot2Stock, focusing on the scenario where a criminal wants to manipulate a stock market using a botnet of infected victims. We consider a malware that behaves similarly to banking trojans (e.g., Zeus, GameOver) by infecting a victim's web browser. The way these infections occur (e.g., phishing or software exploitation) is already studied in prior work.

5.1 Hijacking Brokerage Accounts at Scale

We focus on brokerage defenses that prevent or create awareness of account intrusions. We collect our data from 3 of the 6 most popular online brokerages in the United States. Our analysis includes the availability of 2FA and default settings for event alerts. Namely, when they are triggered and where they are delivered. Our findings are summarized in Table 6.

[3] $5 per order is an upper bound for commission fees. In recent years, brokerages have been eliminating them entirely [16].

Table 5: Cost/Benefit Analysis (Bot Perspective) for the Sustainable Design

Round	Capital from Bots	Profit/Loss Per Instance	Instances in Parallel (1,000 bots)	Accumulated Bot Loss	Bot Net Profit
Taub #1 Botmaster Round	\$401, 000	\$3, 285/\$728	12	\$8, 736	\$0
Taub #1 Bot Round	\$901, 000	\$3, 285/\$728	5	\$12, 376	\$4, 049
Taub #2 Botmaster Round	\$303, 000	\$4, 927/\$1, 345	14	\$18, 830	\$0
Taub #2 Bot Round	\$792, 000	\$4, 927/\$1, 345	6	\$26, 900	\$2, 662
Taub #3 Botmaster Round	\$219, 000	\$24, 501/\$3, 065	22	\$67, 430	\$0
Taub #3 Bot Round	\$555, 000	\$24, 501/\$3, 065	9	\$95, 015	\$12, 5494

Table 6: Comparison of Brokerage Security Features

	TOTP 2FA			Alerts	
Brokerage	Software	Hardware	SMS	Email	Mobile
Brokerage A	◐	◐		●	◐
Brokerage B	●	◐		●	◐
Brokerage C	●		●	●	◐

TOTP ◐ = Setup requires call Alert ◐ = Disabled by default

Two-Factor Authentication. By default, the three online brokerages *do not* require 2FA, which is consistent across the industry. Other work has shown that users are unlikely to take the initiative to setup 2FA if it is not mandatory [38], lowering the bar for attacks like phishing.

At the time of writing, all three brokerages support software TOTP via the Symantec VIP Access application, as shown in the left-most column of Table 6. Brokerages A and B also support TOTP via a dedicated hardware token (second column), whereas C uses SMS or phone calls (third column).

While hardware TOTP is regarded as more secure than its software alternative, SMS and phone calls are weaker due to threats like SIM swapping [40], which can allow an adversary to intercept codes. All of these 2FA schemes can be phished or intercepted, which works in the botmaster's favor.

Alert Triggers & Delivery. By default, all three online brokerages generate alerts when security settings are modified and when orders are created or their status changes (e.g., filled or canceled). A summary is also generated at the end of each day where orders occurred. Users can disable the per-order notifications to avoid being bombarded during frequent trading.

By default, all three brokerages deliver their alerts via email only, as shown in the fourth column of Table 6. Users can also receive notifications on their mobile devices (fifth column), but they must enable this feature.

Email notifications can be silently deleted by the malware using filter rules (Subsection 5.2). Mobile notifications can only be deleted at scale if the malware infects the mobile device, presenting a higher risk of discover. However, this is not the default behavior for brokerage accounts.

5.2 Architecture

We base the design of Bot2Stock after the man-in-the-browser architectures typically utilized by banking trojans. Specifically, our proof-of-concept malware is able to: 1) read and modify any web page visited by the user, 2) record all HTTP(S) header information, including cookies, 3) spawn additional browser sessions to perform arbitrary web requests, 4) add and remove filter rules from popular email services (i.e., Google, Yahoo, Microsoft), and 5) create trade orders in popular brokerage services and simulators.[4] We have open sourced our proof-of-concept and recorded a demonstration video of the attack.[5]

Figure 1 shows the steps in performing the attack. First, the malware adds itself as a certificate authority to the victim's browser. It then spawns a local proxy server and configures the browser to route all traffic through it. *SSL bumping* is used so the malware can decipher encrypted data.

The malware's core logic is implemented inside an *internet content adaptation protocol* (ICAP) server. It is able to arbitrarily read and modify requests and spawn and control additional browser sessions.

The malware silently captures credentials and session cookies as the user navigates sites. Once it has the necessary materials to access the user's email and brokerage accounts, a new browser session is spawned with this data. For demonstration purposes, we make this browser visible so researchers can observe the actions being performed by the malware.

Before performing any manipulations, the malware adds filter rules to the victim's email account so trade notifications will be silently deleted. These rules are very flexible, allowing the malware

[4]For ethical reasons, we only open source the simulator automation.
[5]https://github.com/carter-yagemann/Bot2Stock/blob/master/demo.mp4

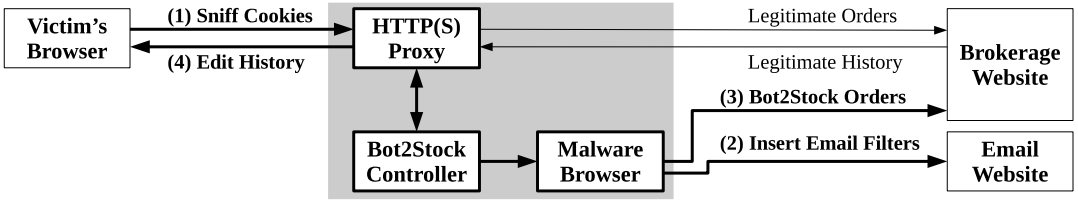

Figure 1: Bot2Stock architecture. Using a proxy and ICAP server, Bot2Stock is able to read, modify, and spoof web traffic.

to delete the brokerage notifications its actions generate while allowing any made by the victim to pass through [21, 32, 33].

The malware then contacts the botmaster and makes its malicious trades while keeping a transaction log. If the victim accesses their history in the brokerage site, the malware will intercept the response and remove the malicious orders to hide them, presenting the user with a falsified transaction history.

6 SIMULATION

The analysis from Section 4 demonstrates that layering is profitable for real-world criminals and may even be self-sustainable for a botnet under the right settings. However, all the studied cases were performed manually by one or two perpetrators. It is unclear whether layering can be used by a botnet to the same result. To address this question, we turn to agent-based simulation to consider the performance of Bot2Stock under various conditions.

6.1 Simulator Design

Stock markets are extremely difficult to model and simulations are hard (if not impossible) to validate for realism. To simulate Bot2Stock, we leverage an *agent-based discrete event* simulator [9], which is the most advance type of simulation accepted for modeling stock markets [28, 46, 52]. It is also used in industry to evaluate multi-agent interactions [5, 34].

Our simulation consists of a collection of *background agents* that trade a stock based on a mixture of trading strategies (elaborated on in Subsection 6.2). Orders are placed by sending messages to an *exchange agent* that maintains an order book for the stock. Messages follow the same protocol used by NASDAQ, which includes allowing agents to query for the latest order stream and current order book spread. Thus, agents can adjust their strategies in reaction to the current state of the open orders at the exchange.

The simulation models time at a nanosecond granularity. Agents are "woken up" by the simulator's kernel when it is their turn to perform computations and send messages. When an agent finishes performing its actions for that time step, it is put back to sleep. The kernel awakens agents as they receive messages and at specific times requested by the agent.

To account for the time it would take a real-world agent to perform its computations, we apply a constant *computational delay* factor whenever an agent wakes up along with a *latency delay*, which is calculated as the sum of a constant minimum latency and a non-negative random noise factor:

$$a + b_{(i,j)} + P(i,j) \tag{1}$$

Here, a is the computational delay constant, $b_{(i,j)}$ is the minimum latency from agent i to agent j and $P(i,j)$ is a random noise factor for that connection.

6.2 Background Agents

Similar to related work on simulating stock markets, we use a combination of *Zero Intelligence* (ZI) and *Heuristic Belief Learning* (HBL) agents to represent the benign traders. Both agents rely on a fundamental belief value for the worth of the stock, which they derive from noisy observations provided by an oracle. At the start of the simulation, these agents enter the market with a Poisson distribution and place their orders based on their trading strategies, which we elaborate on in the following paragraphs. Readers interested in the exact formulas should refer to the spoofing work by Wang and Wellman [51].

Zero Intelligence (ZI). These agents randomly buy and sell shares based on the current price of the stock and their fundamental belief, which they regularly observe from the oracle. More specifically, the decision to buy or sell is picked randomly and the limit price is a bounded uniformly random offset from the fundamental belief value. There is also a strategic threshold, which allows the ZI agent to place an order at the current price if it is within a certain threshold of the fundamental belief.

Heuristic Belief Learning (HBL). These agents start with the same strategy as the ZI agents, but also track the stream of recent orders up to a configured memory length. Once enough orders exist to fill the memory, the HBL agents start adjusting the limit prices of their orders based on the transacted and rejected bids and asks. In other words, unlike the ZI agents, these agents are influenced by order book pressure, which is necessary to model the impact of layering.

Fundamental Belief Oracle (FBO). In order for the agents to create reasonable trades, they each need a fundamental belief of what the stock is worth. To control these fundamental belief values, we use a special agent to act as an oracle. This agent has no computational delay or network latency and when agents query it to update their beliefs, they receive a noisy reading of the fundamental value of the stock at that particular time step. Prior work has shown that such an oracle can be used to guide the simulation in following the trading patterns of real historical data [9], which to our knowledge is the best evidence to date presented for defending the simulation validity of a stock market simulator.

To avoid adding unnecessary complexity to our results, we use a *mean reverting* FBO, which maintains a constant fundamental belief value prior to adding noise for particular observations by the

background agents. With this oracle, the background agents will tend to drive the stock price towards a mean value in the absence of manipulation.

6.3 Bot2Stock Agents & C2 Protocol

Our botmaster's strategy is as follows: 1) buy shares when the market opens at the best available price, 2) wait for a predetermined attack time, 3) signal the layering bots (i.e., hijacked accounts) to begin their manipulation, 4) wait a predetermined duration, 5) sell the previously acquired shares, and then 6) signal the bots to cease manipulation. Note that in a real-world setting, the botmaster would more likely buy shares slowly over an extended period of time to reduce the risk of being detected, but for simplicity we reduce the pre-attack setup to a single bulk buy.

For the layering bots, our agents periodically poll the botmaster and wait quietly for the attack signal. When it is raised, the bots follow the layering strategy described in Subsection 4.1. They periodically poll the exchange to track the order book spread while placing and canceling orders accordingly to maintain open bids that are always slightly worse than the current best bid. For simplicity, we only simulate the bots placing *bids* to drive the price *up*, but it is feasible for them to also place *asks* to *drop* the price. When they receive the signal to stop, they cancel all remaining open orders.

In the ideal scenario, the bot orders would never execute because they are always slightly worse than the best available offer. However in practice, due to delays and latency, some orders will execute by accident, which negatively impacts the botmaster's profits by causing the price to move in the undesired direction. Thus, one of the key factors in determining the botnet's success is in maintaining open orders close enough to the best offer to influence the HBL agents while avoiding executing too many by accident.

6.4 Evaluation Methodology

The purpose of our experiments is to evaluate the robustness of Bot2Stock under different settings. Each experiment consists of several hundred trial pairs for each tested value of the independent variable. Pairs consist of a "control" simulation with no bot agents and a "treatment" simulation with bots. The same random seed is used within each pair so the background agents will make the same decisions, thus isolating the impact caused by the bots. The dependent variable we measure is the difference in the botmaster's profits with and without the bots.

Each trial simulates 2.5 seconds, which we find is enough time for the layering to have an impact on the stock price. We use a conservative computational delay of 10 milliseconds and a Poisson distribution function for P in Equation 1. The background agents consist of 49 ZI and 16 HBL agents with parameters that match a related work on modeling spoofing [51].

For the evaluation, we perform two different experiments. In the first, we vary the ratio of bot to background agents and measure the impact on the botmaster's profits. In the second, we steadily increase the latency between the bots, the exchange, and the botmaster, starting with the same latency as the background agents.

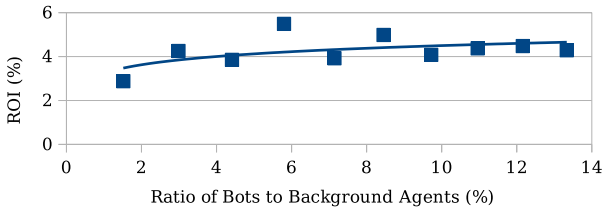

Figure 2: The ROI of Bot2Stock relative to the ratio of bots to background trading agents. The ROI increases slightly as the number of bots increase with a minimum ROI of 2.8% at a 1.5% "bot to background agent" ratio.

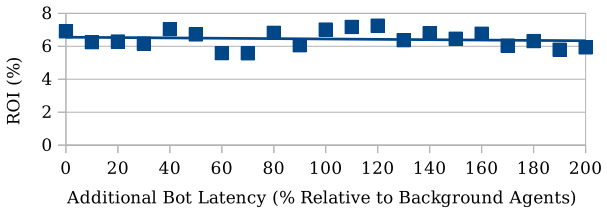

Figure 3: ROI of Bot2Stock relative to network latency of bots. Latency is shown relative to background agents for one direction, with 0% being identical latency. Even at 200% additional latency, the attack remains stable.

6.5 Experimental Results

Our simulations show that the minimum ROI for the botmaster is 2.8% over 2.5 seconds, assuming a 1.5% ratio of bots to background traders. The ROI remains stable even when bots have 200% more network latency than background traders.

Annual Return on Investment. To calculate a conservative annual ROI for the botmaster, we assume 1 attack is carried out per trading day to give the botmaster ample time to perform the pre-attack setup. Assuming 252 trading days in a year (365 days minus 104 weekend days and 9 U.S. public holidays), the botmaster's non-compounding ROI is $252r$ where r is the ROI for a single attack. We use non-compounding ROI to be conservative, although realistically a criminal is likely to reinvest their earnings, resulting in higher profits. Given our minimum estimated ROI of 2.8%, the botmaster would achieve a 1,022% annual non-compounding ROI. In other words, if the botmaster started with $100,000, he would have $1,022,000 after a year. This matches the order of magnitude of the real layering fraud prosecuted by the SEC, which we presented in Section 4.1.

Required Number of Bots. How does a 1.5% ratio of bots translate into real-world market environments? In March 2020, IBM (a "large cap" stock [12]) had an average *minutely* trade volume of 9,120 shares[6], worth $1,241,141 at its highest price that month of $136.10 per share [14]. To sustain 1.5% of this volume for 1 minute,

[6] 3,556,538 shares per day, 22 trading days in March 2020, the NYSE is open from 9:30 AM EST to 4 PM (6.5 hours).

Table 7: Comparison of Monetization Schemes

Crime Type	Annual Revenue	# Bots	Source
DDoS-for-hire	$312,000	N/A	[8]
Spam	$3,500,000	52,000	[23]
Pharmaceutical	$42,500,000	N/A	[30]
Bank Trojan	$302,000,000	180,000	[48]
Ad Fraud	$1,000,000,000	2,000	[3]
Bot2Stock	**$1,022,000**	**~1,000**	

the botmaster would only need $18,617. Assuming each victim has $5,000 in cash (Subsection 4.3), 4 bots are required per attack. To run a yearlong campaign, conducting 1 attack per trading day (252 days) without ever reusing a bot, about 1,000 bots are required, *cumulatively*. Since each attack takes less than a minute, no bot would need to be controlled for more than a day. This is less than the number of accounts our dark web scraper observed being sold in a 3 month window for 1 brokerage on 1 marketplace.

Impact of Bot Ratio. Figure 2 shows the impact to the botmaster's ROI of changing the ratio of bots to background agents. As the number of bots increase, the ROI also increases. In the worst case, with a 1.5% ratio of bots to background agents, the ROI is 2.8% over 2.5 seconds. The bots lost no more than 0.18% of their cash from accidental executions, which is consistent with the loss estimated in Section 4.

Impact of Network Latency. Figure 3 shows the impact of network latency on the stability of Bot2Stock. Latency (x-axis) is shown relative to the latency of the background agents *for one direction*, with 0% denoting identical latency. Note that querying the exchange requires a full round-trip, so total additional latency for round-trip time (RTT) is doubled. Surprisingly, even with 200% additional latency, the layering remains effective. On closer examination, we discover that because background agents wait for an order confirmation from the exchange before issuing another order and some orders never end up executing, the rate at which orders execute at the exchange remains slow enough to render the additional bot latency moot. For example, when background agents have 10 ms network latency, trades execute every 30 ms, on average. Thus, even if the bots have 30 ms latency (200% added), they can still keep up with the price movement.

7 DISCUSSION

7.1 Limitations

We resort to using a simulator to evaluate Bot2Stock because conducting experiments in a real-world marketplace would be highly unethical (and illegal). Consequently, the validity of our results relies on the realism of the simulation. Agent-based discrete event simulations are currently the most realistic technique for modeling stock markets [5, 34, 51], however it is impossible to formally prove that any models accurately reflect real-world markets.

Due to the proprietary and opaque nature of commercial anomaly detection software, we are unable to directly evaluate Bot2Stock's ability to evade real-world detection. However, the SEC case files reveal some clues as to the patterns these programs look for, which we have shown are not produced by Bot2Stock (e.g., Q6 in Subsection 4.1). We also discuss academic systems in Section 2 and why they are currently insufficient. The largest challenge, which works in the criminal's favor, is the high cost of false positives.

Lastly, since the Bot2Stock malware relies on its man-in-the-browser position to hide injected orders from the transaction history page, if the user uses multiple devices to access their brokerage account, they may be able to spot the attack. However, as we show in our evaluation, attacks are conducted so quickly that the user is unlikely to spot the discrepancy in time to prevent it.

7.2 Profit Compared to Other Botnet Schemes

How does the profit of a Bot2Stock malware compare to existing botnet schemes like spam and click fraud? Given the illicit nature of these activities, estimates for current botnet profits are rough and varied. Likewise, since no criminals have been caught automating stock market fraud, our estimates are based on the outcome of discovered manual manipulations and our simulations from Section 6.

Based on prior work, we conclude that the potential profits of a Bot2Stock malware are in the same order of magnitude. For example, McCoy et al. analyzed a four-year pharmaceutical scam generating $42.5 million per year [30]. On the high end of the spectrum, recent analysis by Anderson et al. [3] suggests that an advertising fraud campaign busted by the FBI may have been generating slightly over $1 *billion* of revenue per year, but the authors acknowledge that the lack of public data makes their estimation weak.

It is even harder to estimate the number of bots in a campaign because unless authorities can gain access to the C2 infrastructure, the only observable outcome is the damage caused to victims (e.g. the number of stolen credit card numbers, spam emails sent, etc.). Even when access is gained into the infrastructure, estimates are difficult to make due to churn caused by new infections, old ones being cleaned up, etc. Given the prior work, a safe estimate for the size of real-world botnets is in the order of thousands.

In summary, real-world botnet operations are roughly estimated to make millions per year using thousands of bots. By comparison, our simulation results show that Bot2Stock can make $1,022,000 per year, which is in the same order of magnitude. Table 7 summarizes our comparisons. The revenues are for single campaigns and botnet sizes (where available) are at the time the network was dismantled (i.e., not cumulative). Since cumulative size estimates are unavailable, the reported revenues may be lower bounds. Regardless, even with conservative size estimates, all the botnets in the table are larger than the cumulative size we estimate for operating Bot2Stock for 1 year (Subsection 6.5). Bot2Stock also has the additional advantage of only needing to control each bot for less than a day, due to the speed of layering attacks. That said, we do not know the the ratio of infected devices that access suitable brokerage accounts, however our dark web data suggests enough accounts do get hijacked to conduct a Bot2Stock attack (Subsection 4.3).

7.3 Towards Defending Against Bot2Stock

Increasing Transparency & Accountability. The finest level of data available to traders is layers. Since they aggregate all open orders at a given price, the trader does not know if a layer is made up of many tiny orders or a few large ones or even how many

traders are involved. *This is one of the key reasons why it is possible for criminals to manipulate the market.*

As for the exchanges, they can see the orders, but they only know the identity of the broker, not the brokerage or trader. This poses a challenge for regulatory bodies like the SEC because if they want to investigate a trading event, they have to ask the exchange to identify the broker, then ask the broker for the brokerage, and then finally ask the brokerage for the customer's identity.

In the coming years, companies will be required to participate in the Consolidated Audit Trail program [15], which aims to optimize the deanonymization process. Initial deployment is still ongoing as of 2020 and the program raises its own security concerns because it creates a centralized treasure trove of highly sensitive financial data (names, SSNs, etc.) [13].

Improving Notifications. Although layering attacks may be too quick to stop individually, since they take less than a minute to perform, providing customers with better notifications can reduce the risk that an intrusion goes undetected for an extended period. First, Bot2Stock exploits that notifications default to email-only, which can be filtered server-side. Also sending notifications to a mobile device via SMS or an application would remove this choke-point. Second, sending alerts for new logins would make it harder for adversaries to scope out victim accounts prior to committing the fraud. However, this will only be effective if the first point is addressed, otherwise these alerts will also be filtered by the criminal.

Securing Accounts. Bot2Stock relies on bots being able to control accounts used for trading. It may be tempting to declare that this can all be solved by mandating the use of 2FA, but unfortunately that would be oversimplifying the problem. Mandating 2FA is already easily within the current capabilities of brokerages and yet they choose not to go down this route. One reason is because the financial industry highly values availability, so they are concerned about customers losing their second factor. As one industry expert we interviewed simply stated, customers panic if they cannot access their money. 2FA is also at odds with algorithmic trading, which accounts for over 85% of market volume [19].

Anomaly Detection. There is also more work to be done in detecting anomalous trading patterns. Unfortunately, as we point out in Section 2, existing proposals struggle when the number of identities performing the manipulation is large. Many of the existing products are still trying to address fraud perpetrated by one or two identities, let alone a distributed scheme like Bot2Stock. It is also unclear how the different network layers can benefit detection. For example, brokerages are in a prime position to detect when a particular account's activity abruptly changes (e.g. trading frequency), but lack a complete context across brokerages to draw correlations. Conversely, stock markets lack the per-account activity hidden behind brokers. Thus, a solution will likely require more data sharing between parties, such as the CAT program mentioned earlier.

8 CONCLUSION

This work presents the first study on the feasibility of automating stock market manipulation using a botnet. Our design addresses several key challenges based on data we collected. We determine which techniques are likely to be used by a botmaster, how they can be adapted to a distributed network of bots, how it will evade detection, among other technical challenges.

We implement our design in a proof-of-concept, man-in-the-browser malware, Bot2Stock, and evaluate it using agent-based simulations. We discover that 1,000 bots, cumulatively collected over a yearlong campaign, can yield an average ROI of 1,022% if 1 attack is performed daily, allowing a botmaster to turn $100,000 into $1,022,000. This is consistent with our collected real-world data and comparable to alternative botnet schemes.

ACKNOWLEDGMENTS

The authors would like to thank Yanick Fratantonio for his contributions in the early stages of this work.

REFERENCES

[1] Michael J Aitken, FH Harris, and Shan Ji. 2009. Trade-Based Manipulation and Market Efficiency: A Cross-Market Comparison. In *Proceedings of the 22nd Australia Finance Banking Conference*. SSRN, 55.
[2] Franklin Allen and Douglas Gale. 1992. Stock-Price Manipulation. *The Review of Financial Studies* 5, 3 (1992), 503–529.
[3] Ross Anderson, Chris Barton, Rainer Böhme, Richard Clayton, Michel J. G. van Eeten, Michael Levi, Tyler Moore, and Stefan Savage. 2013. *Measuring the Cost of Cybercrime*. Springer Berlin Heidelberg, Berlin, Heidelberg, 265–300. https://doi.org/10.1007/978-3-642-39498-0_12
[4] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. 2017. Understanding the mirai botnet. In *26th USENIX Security Symposium (USENIX Security 17)*. 1093–1110.
[5] Samuel Assefa, Danial Dervovic, Mahmoud Mahfouz, Tucker Balch, Prashant Reddy, and Manuela Veloso. [n.d.]. Generating synthetic data in finance: opportunities, challenges and pitfalls. ([n. d.]).
[6] Rainer Böhme and Thorsten Holz. 2006. The Effect of Stock Spam on Financial Markets.
[7] Matthew Brand, Nuria Oliver, and Alex Pentland. 1997. Coupled Hidden Markov Models for Complex Action Recognition. In *CVPR*. IEEE, IEEE, 994–999.
[8] Ryan Brunt, Prakhar Pandey, and Damon McCoy. 2017. Booted: An Analysis of a Payment Intervention on a DDoS-for-Hire Service. In *Workshop on the Economics of Information Security*. WEIS, 06–26.
[9] David Byrd, Maria Hybinette, and Tucker Hybinette Balch. 2019. ABIDES: Towards High-Fidelity Market Simulation for AI Research. *arXiv preprint arXiv:1904.12066* (2019), 13.
[10] Longbing Cao, Yuming Ou, Philip S Yu, and Gang Wei. 2010. Detecting Abnormal Coupled Sequences and Sequence Changes in Group-Based Manipulative Trading Behaviors. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM.
[11] Yi Cao, Yuhua Li, Sonya Coleman, Ammar Belatreche, and Thomas Martin McGinnity. 2016. Detecting Wash Trade in Financial Market Using Digraphs and Dynamic Programming. *IEEE transactions on neural networks and learning systems* 27 (2016).
[12] James Chen. [n.d.]. Large Cap (Big Cap). https://www.investopedia.com/terms/l/large-cap.asp. Accessed: 2020-04-17.
[13] Jay Clayton. [n.d.]. Statement on Status of the Consolidated Audit Trail. https://www.sec.gov/news/public-statement/statement-status-consolidated-audit-trail. Accessed: 2020-08-21.
[14] YahooFinance. [n.d.]. International Business Machines Corporation (IBM). https://finance.yahoo.com/quote/IBM/history. Accessed: 2020-04-17.
[15] FINRA. [n.d.]. The Consolidated Audit Trail. https://www.catnmsplan.com/. Accessed: 2019-04-03.
[16] Maggie Fitzgerald. [n.d.]. Charles Schwab is Eliminating Online Commissions for Trading in US Stocks and ETFs. https://www.cnbc.com/2019/10/01/charles-schwab-is-eliminating-online-commissions-for-trading-in-us-stocks-and-etfs.html. Accessed: 2020-04-15.
[17] Laura Frieder and Jonathan Zittrain. 2007. Spam Works: Evidence From Stock Touts and Corresponding Market Activity. *Hastings Comm. & Ent. LJ* (2007).
[18] Eric Gilbert and Karrie Karahalios. 2010. Widespread Worry and the Stock Market. In *Fourth International AAAI Conference on Weblogs and Social Media*.
[19] Morton Glantz and Robert L Kissell. 2013. *Multi-Asset Risk Modeling: Techniques for a Global Economy in an Electronic and Algorithmic Trading Era*. Academic Press.
[20] Koosha Golmohammadi and Osmar R Zaiane. 2015. Time Series Contextual Anomaly Detection for Detecting Market Manipulation in Stock Market. In *2015*

IEEE International Conference on Data Science and Advanced Analytics (DSAA). IEEE.
[21] Google. [n.d.]. Search operators you can use with Gmail. https://support.google.com/mail/answer/7190?hl=en. Accessed: 2020-04-17.
[22] Michael Hanke and Florian Hauser. 2008. On the Effects of Stock Spam E-Mails. *Journal of Financial markets* 11 (2008).
[23] Chris Kanich, Christian Kreibich, Kirill Levchenko, Brandon Enright, Geoffrey M Voelker, Vern Paxson, and Stefan Savage. 2008. Spamalytics: An Empirical Analysis of Spam Marketing Conversion. In *Proceedings of the 15th ACM conference on Computer and communications security*. ACM.
[24] Asim Ijaz Khwaja and Atif Mian. 2005. Unchecked Intermediaries: Price Manipulation In an Emerging Stock Market. *Journal of Financial Economics* 78 (2005).
[25] Albert S Kyle and S Viswanathan. 2008. How to Define Illegal Price Manipulation. *American Economic Review* 98 (2008).
[26] Mark Langston. [n.d.]. How To: Building A Dark Web Scraper. https://justhackerthings.com/post/building-a-dark-web-scraper/. Accessed: 2019-02-04.
[27] Aihua Li, Jiede Wu, and Zhidong Liu. 2017. Market Manipulation Detection Based on Classification Methods. *Procedia Computer Science* 122 (2017).
[28] Junyi Li, Xintong Wang, Yaoyang Lin, Arunesh Sinha, and Michael P Wellman. 2018. Generating Realistic Stock Market Order Streams. (2018).
[29] Yuexin Mao, Wei Wei, Bing Wang, and Benyuan Liu. 2012. Correlating S&P 500 Stocks with Twitter Data. In *Proceedings of the 1st ACM International Workshop on Hot Topics on Interdisciplinary Social Networks Research*. ACM.
[30] Damon McCoy, Andreas Pitsillidis, Jordan Grant, Nicholas Weaver, Christian Kreibich, Brian Krebs, Geoffrey Voelker, Stefan Savage, and Kirill Levchenko. 2012. PharmaLeaks: Understanding the Business of Online Pharmaceutical Affiliate Programs. In *21st USENIX Security Symposium ((USENIX) Security 12)*.
[31] Jianping Mei, Guojun Wu, and Chunsheng Zhou. 2004. Behavior Based Manipulation: Theory and Prosecution Evidence. (2004).
[32] Microsoft. [n.d.]. Mail flow rule actions in Exchange Online. https://docs.microsoft.com/en-us/exchange/security-and-compliance/mail-flow-rules/mail-flow-rule-actions. Accessed: 2020-04-17.
[33] Microsoft. [n.d.]. Mail flow rules (transport rules) in Exchange Online. https://docs.microsoft.com/en-us/exchange/security-and-compliance/mail-flow-rules/mail-flow-rules. Accessed: 2020-04-17.
[34] Vaikkunth Mugunthan, Antigoni Polychroniadou, David Byrd, and Tucker Hybinette Balch. [n.d.]. SMPAI: Secure Multi-Party Computation for Federated Learning. ([n. d.]).
[35] Maithijs Nelemans. 2007. Redefining Trade-Based Market Manipulation. *Val. UL Rev.* 42 (2007).
[36] Karen K Nelson, Richard A Price, and Brian R Rountree. 2009. Why Do Investors Pay Attention to Stock Spam? *Jones Graduate School of Management, Rice University, Houston, TX* 77005 (2009).
[37] Venkata Sasank Pagolu, Kamal Nayan Reddy, Ganapati Panda, and Babita Majhi. 2016. Sentiment Analysis of Twitter Data for Predicting Stock Market Movements. In *2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*. IEEE.
[38] Thanasis Petsas, Giorgos Tsirantonakis, Elias Athanasopoulos, and Sotiris Ioannidis. 2015. Two-Factor Authentication: Is the World Ready?: Quantifying 2FA Adoption. In *Proceedings of the Eighth European Workshop on System Security*. ACM.
[39] CGJ Putman and Lambert JM Nieuwenhuis. 2018. Business model of a botnet. In *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*. IEEE, 441–445.
[40] Mary-Ann Russon. [n.d.]. SIM Swap Fraud: The Multi-Million Pound Security Issue That UK Banks Won't Talk About. https://tinyurl.com/y2ewsduz. Accessed: 2019-03-21.
[41] Cheryl Scarboro. [n.d.]. SEC Charges Two Texas Swindlers In Penny Stock Spam Scam Involving Computer Botnets. https://www.sec.gov/news/press/2007/2007-130.htm. Accessed: 2019-02-04.
[42] Cheryl Scarboro. [n.d.]. SEC Obtains Order Freezing $3 Million in Proceeds of Suspected Foreign-Based Account Intrusion Scheme. https://www.sec.gov/litigation/litreleases/2007/lr20030.htm. Accessed: 2019-02-27.
[43] SEC. [n.d.]. SEC Charges Firms Involved in Layering, Manipulation Schemes. https://www.sec.gov/news/pressrelease/2017-63.html. Accessed: 2020-08-20.
[44] SEC. [n.d.]. SEC Files Charges in $26 Million Stock Manipulation Scheme. https://www.sec.gov/news/pressrelease/2016-261.html. Accessed: 2020-08-21.
[45] SEC. [n.d.]. Trading Basics: Understanding the Different Ways to Buy and Sell Stock. https://www.sec.gov/investor/alerts/trading101basics.pdf. Accessed: 2019-02-04.
[46] Megan Shearer, Gabriel Rauterberg, and Michael P Wellman. 2019. An Agent-Based Model of Financial Benchmark Manipulation. In *ICML-19 Workshop on AI in Finance*.
[47] Yin Song, Longbing Cao, Xindong Wu, Gang Wei, Wu Ye, and Wei Ding. 2012. Coupled Behavior Analysis for Capturing Coupling Relationships in Group-Based Market Manipulations. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM.
[48] Brett Stone-Gross, Marco Cova, Lorenzo Cavallaro, Bob Gilbert, Martin Szydlowski, Richard Kemmerer, Christopher Kruegel, and Giovanni Vigna. 2009. Your Botnet Is My Botnet: Analysis of a Botnet Takeover. In *Proceedings of the 16th ACM conference on Computer and Communications Security (CCS'16)*. ACM.
[49] Trillium. [n.d.]. Recent Trade Surveillance Enforcement Actions. https://www.trlm.com/knowledgebase/recent-trade-surveillance-enforcement-actions. Accessed: 2019-05-04.
[50] Manuel R Vargas, Beatriz SLP De Lima, and Alexandre G Evsukoff. 2017. Deep Learning for Stock Market Prediction From Financial News Articles. In *2017 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*. IEEE.
[51] Xintong Wang and Michael P Wellman. 2017. Spoofing the Limit Order Book: An Agent-Based Model. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems.
[52] Mason Wright and Michael P Wellman. 2018. Evaluating the Stability of Non-Adaptive Trading in Continuous Double Auctions. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems.
[53] Jiahua Xu and Benjamin Livshits. 2019. The anatomy of a cryptocurrency pump-and-dump scheme. In *28th USENIX Security Symposium (USENIX Security 19)*. 1609–1625.

A APPENDIX

Layering is tricky to grasp because of the indirect way pressure from open orders drives stock prices. To help visualize it, we list the *non-bona fide* trades made in one real-world instance in Table 8. As shown, even though only half the orders actually executed (the rest were cancelled by the criminals), the stock price moved 0.66% in a matter of minutes. If, for example, the criminal bought $1,000 worth of shares prior to this manipulation, he could sell them immediately following the layering for $1006.60. While this gain may seem small, with enough leverage (i.e., shares bought in advance) and consistency, criminals can turn even this seemingly indiscernible movement into a profitable scam.

Table 8: One instance of non-bona fide orders reported in SEC v.s. Taub et al. In a few minutes, the criminal increased the stock's price by 0.66%, despite most of the orders being canceled. By following these orders with a bona fide ask, the adversary can cash out any shares he purchased in advance at the elevated price.

Order Type	# Shares	# Executed	Price	Capital Needed
Buy	500	0	$69.48	$34, 740
Buy	100	100	$69.68	$6, 968
Buy	400	0	$69.65	$27, 860
Buy	100	100	$69.77	$6, 977
Buy	100	100	$69.80	$6, 980
Buy	100	0	$69.79	$6, 979
Buy	100	0	$69.79	$6, 979
Buy	100	100	$69.84	$6, 984
Buy	100	100	$69.85	$6, 985
Buy	100	0	$69.83	$6, 983
Buy	100	0	$69.85	$6, 985
Buy	100	100	$69.93	$6, 993
Buy	100	100	$69.93	$6, 993
Buy	100	100	$69.95	$6, 995
Buy	100	0	$69.88	$6, 988
Buy	100	0	$69.91	$6, 991
Buy	100	100	$69.94	$6, 994
Buy	100	100	$69.98	$6, 980
Buy	100	100	$69.94	$6, 994
Total	2,600	1,100	Δ0.66%	$184, 066

To simplify the following tables, we use the notation (b, s, x) to denote buying x shares at price b, sold at price s.

Table 9: Manipulative Orders made in Taub #1 example (price defaltion part, including commission fees)

Order	Executed?	Cost	Capital Needed
sell 100@$69.69	yes	$9	$6, 973
sell 100@$69.77	yes	$1	$6, 973
sell 1000@$69.69	yes	$90	$69, 730
($69.69, $69.60, 100)	yes	$14	$6, 969
($69.77, $69.60, 100)	no	$0	$6, 977
($69.60, $69.57, 100)	yes	$8	$6, 960
($69.57, $69.53, 100)	yes	$9	$6, 957
($69.53, $69.50, 100)	yes	$8	$6, 953
sell 900@$69.50	no	$0	$62, 757
($69.50, $69.42, 100)	no	$0	$6, 950
sell 100@$69.49	yes	$29	$6, 973
($69.49, $69.46, 100)	yes	$8	$6, 949
($69.46, $69.44, 100)	no	$0	$6, 946
sell 100@$69.46	yes	$42	$6, 973
total		$218	$217, 036

Table 10: Manipulative Orders made in Taub #2 example (price defaltion part, including commission fees)

Order	Executed?	Cost	Capital Needed
sell 200@$79.59	yes	$35	$15, 938
($79.59, $79.35, 200)	yes	$63	$15, 918
sell 100@$79.52	yes	$27	$7, 969
($79.52, $79.47, 100)	yes	$15	$7, 952
($79.47, $79.37, 100)	yes	$20	$7, 947
($79.37, $79.27, 100)	yes	$20	$7, 937
sell 100@$79.41	yes	$65	$7, 969
sell 100@$79.80	yes	−$1	$7, 969
($79.80, $79.73, 100)	yes	$17	$7, 980
sell 100@$79.70	yes	$9	$7, 969
total		$270	$95, 548

Table 11: Manipulative Orders made in Taub #2 example (price infaltion part, including commission fees)

Order	Executed?	Cost	Capital Needed
buy 100@$79.73	yes	$47	$7, 973
buy 100@$79.73	yes	$47	$7, 973
buy 100@$79.73	yes	$47	$7, 973
buy 100@$79.73	yes	$47	$7, 973
buy 600@$79.66	no	$0	$7, 966
buy 200@$79.77	no	$0	$7, 977
buy 300@$79.90	yes	$187	$23, 970
buy 100@$79.77	no	$0	$7, 977
buy 100@$79.98	yes	$72	$7, 998
buy 100@$80.00	yes	$74	$8, 000
buy 100@$80.00	yes	$74	$8, 000
buy 100@$79.84	no	$0	$7, 984
buy 100@$79.84	no	$0	$7, 984
buy 100@$79.98	yes	$72	$7, 998
buy 100@$80.09	yes	$83	$8, 009
buy 100@$79.88	no	$0	$7, 988
buy 100@$79.88	no	$0	$7, 988
buy 100@$80.05	no	$0	$8, 005
buy 100@$80.42	yes	$116	$8, 042
buy 100@$80.09	no	$0	$8, 009
buy 100@$80.41	yes	$115	$8, 041
buy 100@$80.20	yes	$94	$8, 020
buy 100@$80.27	no	$0	$8, 027
buy 100@$80.37	no	$0	$8, 037
total		$1, 075	$207, 912

Table 12: Manipulative Orders made in Taub #3 example (price deflation part, including commission fees)

Order	Executed?	Cost	Capital Needed
sell 100@$20.03	yes	$12	$2, 010
($20.03, $19.92, 100)	yes	$16	$2, 003
($19.92, $19.89, 100)	yes	$8	$1, 992
($19.89, $19.81, 100)	yes	$13	$1, 989
($19.81, $19.77, 100)	yes	$9	$1, 981
($19.77, $19.73, 100)	yes	$9	$1, 977
($19.73, $19.67, 100)	yes	$11	$1, 973
($19.67, $19.46, 100)	yes	$26	$1, 967
sell 100@$19.46	yes	$69	$1, 946
sell 100@$19.64	yes	$51	$2, 010
($19.64, $19.60, 100)	yes	$11	$1, 964
($19.60, $19.47, 100)	yes	$18	$1, 960
($19.47, $19.46, 100)	yes	$6	$1, 947
sell 100@$19.55	yes	$60	$2, 010
($19.55, $19.46, 100)	yes	$14	$1, 955
sell 100@$19.51	yes	$64	$2, 010
sell 100@$19.52	yes	$63	$2, 010
($19.52, $19.46, 100)	yes	$11	$1, 952
sell 100@$19.49	yes	$66	$2, 010
($19.49, $19.46, 100)	yes	$8	$1, 949
sell 100@$19.52	yes	$63	$2, 010
total		$608	$41, 625

Table 13: Manipulative Orders made in Taub #3 example (price inflation part, including commission fees)

Order	Executed?	Cost	Capital Needed
buy 1500@$20.10	no	$0	$30, 150
buy 1500@$20.44	(200)	$149	$4, 088
buy 100@$20.49	no	$0	$2, 049
buy 100@$20.52	no	$0	$2, 052
buy 1500@$20.57	no	$0	$30, 855
buy 100@$20.60	no	$0	$2, 060
buy 100@$20.63	no	$0	$2, 063
buy 100@$20.66	no	$0	$2, 066
buy 100@$20.81	no	$0	$2, 081
buy 100@$20.84	(45)	$56	$937
buy 100@$20.84	no	$0	$2, 084
buy 100@$20.87	no	$0	$2, 087
buy 100@$20.90	no	$0	$2, 090
buy 100@$21.05	yes	$138	$2, 105
buy 100@$21.05	yes	$138	$2, 105
buy 100@$21.05	yes	$138	$2, 105
buy 100@$20.93	no	$0	$2, 093
buy 100@$20.96	no	$0	$2, 096
buy 100@$20.99	no	$0	$2, 099
buy 100@$21.02	no	$0	$2, 102
buy 100@$21.12	no	$0	$2, 112
buy 1000@$21.09	no	$0	$21, 090
buy 100@$21.15	no	$0	$2, 115
buy 100@$21.24	no	$0	$2, 124
buy 100@$21.27	no	$0	$2, 127
buy 100@$21.29	no	$0	$2, 129
buy 100@$21.30	no	$0	$2, 130
buy 100@$21.33	yes	$166	$2, 133
buy 100@$21.36	yes	$169	$2, 136
buy 100@$21.42	yes	$175	$2, 142
buy 1000@$21.32	yes	$165	$21, 320
buy 100@$21.36	yes	$169	$2, 136
buy 100@$20.86	yes	$119	$2, 086
buy 100@$21.01	yes	$134	$2, 101
buy 100@$21.08	yes	$141	$2, 108
buy 100@$21.14	yes	$147	$2, 114
buy 100@$21.17	yes	$150	$2, 117
buy 100@$21.29	yes	$162	$2, 129
buy 100@$21.08	yes	$141	$2, 108
total		$2, 457	$177, 824