Putting Out a HIT: Crowdsourcing Malware Installs

Chris Kanich
UC San Diego

Stephen Checkoway
UC San Diego

Keaton Mowery
UC San Diego

Abstract

Today, several actors within the Internet's burgeoning underground economy specialize in providing services to like-minded criminals. At the same time, gray and white markets exist for services on the Internet providing reasonably similar products. In this paper we explore a hypothetical arbitrage between these two markets by purchasing "Human Intelligence" on Amazon's Mechanical Turk service, determining the vulnerability of and cost to compromise the computers being used by the humans to provide this service, and estimating the underground value of the computers which are vulnerable to exploitation. We show that it is economically feasible for an attacker to purchase access to high value hosts via Mechanical Turk, compromise the subset with unpatched, vulnerable browser plugins, and sell access to these hosts via Pay-Per-Install programs for a tidy profit. We also present supplementary statistics gathered regarding Mechanical Turk workers' browser security, antivirus usage, and willingness to run arbitrary programs in exchange for a small monetary reward.

1 Introduction

When considering the improvement of business processes, a common trope is that of "cutting out the middleman." Imagine a CEO — or whatever passes for a CEO in the underground world of Internet crime — of a Pay-Per-Install (PPI) affiliate brainstorming new infection vectors for high-value hosts. "Ah ha!" the CEO thinks, "I'll just pay the victims to infect themselves."

The world of ecrime has undergone a radical shift in recent years from one of vertical integration — where a single entity would produce malware, compromise hosts, install the malware on victim machines, and finally extract value, for example via online banking fraud — to one in which each link in the chain from malware to monetization is performed by a different actor [13]. A modern business model relying upon installing malware on compromised machines would involve paying a PPI service such as Loaderadv or Goldinstall for access to previously compromised hosts. The PPI service in turn pays affiliates to install special *downloader* programs on compromised hosts which will download and execute clients' malware. The complete situation is more complex still with some PPI services acting as affiliates for competing services and other arbitrage possibilities. This ecosystem is well analyzed in Caballero et al. [7].

Here we consider the position of a PPI affiliate who is paid to install "downloader" programs on behalf of one or more PPI services. Rather than relying on iframes inserted into compromised websites, our hypothetical PPI affiliate entices workers on Amazon's Mechanical Turk to visit our website using the incentive of a small reward.

Mechanical Turk is a popular "crowdsourcing" service where *requesters* can post jobs, called Human Intelligence Tasks (HITs) which *workers* can choose to perform in exchange for a monetary reward from the requester. Before accepting a HIT, the worker will usually preview the HIT for an example of the task they are being paid to perform.

Returning briefly to our hypothetical PPI affiliate, in order to infect machines using Mechanical Turk, it posts HITs with a description that appears profitable enough for workers to visit the preview page or even perform the HIT. Once the worker visits the page, the affiliate has a number of choices for infecting the worker's computer. The webpage could exploit vulnerabilities in the worker's browser and plugins to initiate a drive-by download or even ask the worker to download and run an executable on their computer as part of the HIT.

Ultimately, we wish to discover the hypothetical cost to install a PPI service's downloader onto a victim machine directly. If this can be done cheaply enough that the amount earned by running one or more downloaders exceeds the costs, then using Mechanical Turk as an infection vector is a viable choice.

We find that using Mechanical Turk to take over machines is economically feasible for high-value machines in the U.S. and possibly feasible for low-value machines.

2 Related Work

One of the main draws in using Mechanical Turk is that human workers can quickly perform tasks that are difficult for computers but easy for humans. This has led to a number of studies where humans are used to provide ground truth for machine learning or vision algorithms [10, 12, 28]. In addition, Mechanical Turk is also popular with social science researchers as a source of data. See, for example, Ruvolo et al. [24], Paolacci et al. [21], and the references therein for a flavor of the sorts of research conducted using Mechanical Turk. Of course, with so many researchers turning to Mechanical Turk, it is natural that Mechanical Turk itself and crowdsourcing more generally would become a topic of study, both for the academic community and for the mainstream press [16, 22, 23, 25].

The most comprehensive source of information on the economics of the Pay-Per-Install is by Caballero et al. [7] who discuss all facets of the PPI ecosystem from the malware that clients want installed to the tools used to avoid detection. Earlier work by Stevens [26] gives examples of the various tools used by PPI affiliates such as "crypters" which try to hide malware from antivirus programs as well as describing the largest PPI brands such as Earning4u.com. The website `http://pay-per-install.com` contains reviews of PPI affiliate programs with what are ostensibly comments by these services' users. However, some of the reviewed services seem like scams offering more than the market rate per install. Many of the comments for these services give supporting evidence that they are indeed scams.

There is evidence that Mechanical Turk is being used as a method of installing adware or spyware on workers computers. StopMalvertising reports [19] that some HITs require workers to fill out CPALead surveys for a 50¢ reward. As part of the survey spyware or adware is installed on the worker's computer. And, in the end, the worker is denied the reward. Such an infection is far more visible than our proposed method of exploiting browser plugins as workers are required to install browser plugins to complete the survey, thus alerting the worker to possible danger.

The works most closely related to ours are a study by Christin et al. [9] in which the authors pay Mechanical Turk workers between 1¢ and \$1 to run an executable; Wondracek et al.'s study of the online adult industry [29] including the susceptibility of adult content viewers' web browser plugins; and Acer and Jackson's preliminary report on using ad networks to evaluate browser vulnerability [1].

Christin et al. describe the executable workers run as a distributed computing client for use as part of a CMU research experiment. The authors find that between 22% and 43% of the workers who viewed the HIT ran the executable. In the present work, we focus on the question of what percentage of workers had vulnerable browser plugins; however, we additionally collect information on the percentage of workers who are willing to download and run code for an additional bonus.[1] Our results which agree with Christin et al.'s are given in Section 5.3.

Wondracek et al. operate adult websites and buy traffic from web traffic brokers. Similar to our work, they evaluate browser plugin vulnerability and report that 88% of the browsers they were able to profile were potentially vulnerable. Our analysis focuses on a different population and finds very similar overall statistics; these are shown along with temporal uptake data in Section 5.2.

Although neither adult web site viewers' browsers nor Mechanical Turk workers' browsers are likely to be representative of Internet users' browsers in general, these studies in conjunction with Acer and Jackson's results, give evidence that the majority of Internet users' browsers are at risk. Evaluating a more representative sample of the Internet is left to future work.

This study uses the same mechanism for plugin enumeration as the Electronic Frontier Foundation's Panopticlick project [11]. Whereas Panopticlick aims to fingerprint individual browser installations, the main pertinent information available to us is the installed plugins' versions, and only the subset for which exploits are known.

Finally, there is a large body of work involving methods of driving traffic to a website, for example by buying ads [2, 6, 15, 18] or other methods of directly purchasing click traffic [30].

3 Methodology

To test the efficacy and economic feasibility of using Mechanical Turk, we post a simple-to-complete HIT to Mechanical Turk for small rewards, 1¢ or 5¢, for several days each. As described in Sections 3.1 and 4, we determine what fraction of workers are vulnerable to well-known exploitable vulnerabilities in plugins such as Adobe Flash and Apple QuickTime. As a bonus, we gain some idea of what percentage of workers are willing to download and run programs for a small additional reward as well as whether workers use antivirus programs, and if their virus definitions are up to date. Finally, we compare the cost to acquire hosts via this strategy with bid prices for Pay-Per-Install affiliate programs to determine economic viability.

3.1 The HITs

On Mechanical Turk, Human Intelligence Tasks range in complexity from answering a simple question to transcribing an hour long podcast. The price of the HITs varies roughly proportionally with the difficulty of the task, with easy tasks paying only 1¢ and more complex tasks pay-

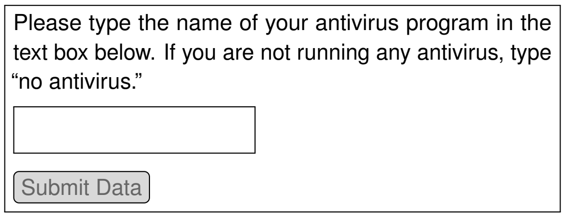
[1] As described in Section 6.4, requiring workers to run executables violates Mechanical Turk's policies [3] and thus we made it optional.

ing as much as $10. Additionally, bonuses can be paid to workers who perform exceptionally well or do extra work.

Amazon provides basic data entry and survey answering services within a HIT creation page available to requesters. Amazon hosts both the pages asking for user input and the back-end recording of these results. Alternately, a requester can create an External Question which loads a web page within an `iframe` and allows the requester to load arbitrary content in the service of completing the HIT. When the worker finishes the task, the requester's page sends a POST message to Amazon's servers indicating completion. In our experiments, we use this feature to both measure the worker's browser and solicit survey answers.

In the usual case, before workers accept a HIT, they view a preview of the task. This preview is just the normal HIT page loaded from the requester's webserver with a fixed `assignmentId` query parameter and displayed in an `iframe` in the worker's browser.[2]

Our HITs fall into the extremely simple category. The HITs we use to collect vulnerability data are shown below.


Please type the name of your antivirus program in the text box below. If you are not running any antivirus, type "no antivirus."

Once the HIT is accepted, the task page includes this additional text:

> **For a bonus of 11 cents**, we can also collect additional information about your antivirus if you download and run this script. This script does not harm or change your computer in any way. You may inspect the script to verify this. After the script has run, a Notepad window will pop up including information about your running antivirus. COPY and PASTE everything in the Notepad window into the text box below.

and a text box to receive the requested data. The bonus is chosen uniformly at random between 1¢ and 15¢ to determine if there is any correlation between the bonus reward and the willingness to run programs outside of the browser.

In addition to the survey HIT, we constructed a second HIT which was as enticing and easy as possible in order to maximize the number of workers who complete a low-reward task. This HIT asks the user to click an HTML form button as quickly as possible for five seconds, recording the number of clicks and paying the worker immediately for their time. We emphasize the ease and speed of the HIT within its description, as well as the fact that all workers will be paid immediately for their efforts. This HIT pays 1¢ and does not collect any browser or plugin information.

To maximize the number of workers who complete our HITs, we follow the recommendations from Chilton et al. in crafting our task [8]. Most notably, the titles "␣Anti Virus Survey" and "␣5 second reaction test! Auto-approve!" include a leading space in order to appear at the beginning of an alphabetical ordering, we skip the 2¢ price point as it was shown to be less popular than 1¢, and the HITs have a very short completion time limit to appear at the top of a time limit ordering (shortest first).

In order to test the effect of the base amount of money the workers receive upon successful completion of the survey HIT on the number of workers who view and accept the HIT, we tested the HIT with a reward of 1¢ and 5¢. Only a single HIT was active at a time and we started with 1¢ before moving to 5¢. We allowed workers to complete both of the HITs if they so chose, although we only offered a bonus if they had not previously received one.

3.2 Vulnerability data collection

We collect information from three sources: (1) JavaScript run in the worker's browser which POSTs the results back to our server immediately; (2) the worker's report of their antivirus; and, for a bonus, (3) the output of a program run by the worker outside the browser context. The first happens automatically whereas the second and third require the worker to enter text.

As soon as the HIT is accepted, JavaScript collects information about the worker's browser and the plugins that are enabled. We use the PluginDetect JavaScript library [14] to determine the operating system and browser version as well as the versions of Adobe Flash, Adobe Reader, Adobe Shockwave, DevalVR, Java, QuickTime, RealPlayer, Microsoft Silverlight, VLC, and Windows Media Player.

As soon as this information is collected it is sent back to the webserver via an `XMLHttpRequest`. This happens without any user action beyond visiting the web page. Indeed, this information could be harvested as soon as a worker visits the preview page. Were an attacker attempting to infect a machine by exploiting a flaw in a browser or a plugin, this is an ideal place to do so since workers usually visit the preview page even if they do not ultimately choose to accept the HIT. As discussed in Section 6.4, we choose not to collect any information from the preview page to avoid collecting information about workers' browsers without their consent.

The second source of information gathered is the name of the antivirus software workers have installed. The workers enter the name of their antivirus, if any, into the

[2] We discovered a very small number of cases where a worker managed to accept a HIT without previewing it. We suspect this is due to a modified Mechanical Turk worker interface such as those provided by some Greasemonkey scripts.

Table 1: Plugin vulnerability by OS. Unless a range is specified, lower versions are also vulnerable.

Plugin	Windows	Mac OS X	Linux	CVE
Adobe Flash Player	10.2.154.13	10.2.154.13	10.2.154.13	CVE-2011-0609
Adobe Reader*	10.0.2	10.0.2	9.4.1	CVE-2011-0610 CVE-2011-0611
Adobe Shockwave Player	11.5.9.615	11.5.9.615		CVE-2011-0557
Apple QuickTime	7.6.8	7.6.8		CVE-2010-3787
Microsoft Silverlight	3.0.50106.0	3.0.40818.0		CVE-2010-1898
Java†	1.6.0–1.6.0.21	1.6.0–1.6.0.21	1.6.0–1.6.0.21	CVE-2010-3571
RealPlayer‡	11.0–11.1	11.0–11.1	11.0.2.1744	CVE-2010-4397
VLC media player	1.1.7			CVE-2010-3276

*CVE-2011-0610 affects all three for versions up to 9.4.1; CVE-2011-0611 only affects Windows and Mac OS X.
†CVE-2010-3571 affects many versions of Java 1.4 and 1.5.
‡Many other versions are affected by different vulnerabilities.

text box. This is the least reliable source of information as there is no incentive for the worker to bother entering an accurate value since we are willing to pay for "no antivirus."[3] Despite this, a majority of the users who ran the bonus program described below reported the same AV as did the program.

The third source of information is optional and garners the workers a bonus reward. To receive the bonus, the worker downloads and runs the program linked from the description of the bonus. The program is written in JScript — a dialect of ECMAScript — and is run by the Windows Script Host. It uses the Windows Management Instrumentation (WMI) interface to enumerate the antivirus products installed. For each AV product installed, the program checks if the virus definition files are up to date or not. For Windows XP and Vista, this is a simple matter of examining the `productUptoDate` property of the enumerated AV products. For Windows 7, this property is no longer exposed; however, the undocumented `productState` integer property has bit 4 set if the virus definitions are not up to date and clear if they are. The specific cutoff for whether or not an antivirus product is up to date is set by the individual vendors. The AV product name, version (if available), and definition status are written to a temporary file. This temporary file is opened in Notepad and the worker then copies and pastes the contents of the file into the text box in the HIT's page.

4 Plugin Vulnerability Information

One measure of vulnerability is to examine the version numbers of plugins and compare them with known vulnerabilities. On the one hand, this is a fairly coarse measure as some software's version numbers do not change when they are patched. For example, Microsoft Internet Explorer and Windows Media Player do not change version numbers when patched. For this reason, we chose not to include vulnerabilities for these in our analysis.

[3]In actuality, we paid the workers for anything entered in this box.

Table 1 lists some of the vulnerabilities we considered when deciding if a particular worker was vulnerable. For each, we list one of the Common Vulnerabilities and Exposures (CVE) entries we used to determine if a version is vulnerable [5].[4] We only consider a plugin version to be exploitable if a CVE lists remote execution of attacker code.

5 Results

Of primary importance to someone looking to compromise hosts using Mechanical Turk are the cost per thousand hosts and the rate at which they can be compromised. In this section, we present the results of our study and answer those questions. In the next section we discuss the ramifications of these results.

The Mechanical Turk population is mostly located in the United States and in India [16, 21, 23]. Indeed, our experiments with Mechanical Turk show roughly similar distributions of workers: 61.3% in India, 23.2% in the U.S. with the remaining 15.5% spread among 75 other countries. Therefore, all of our results in this section are separated into U.S., India, and Other categories.

5.1 Infection cost

In order for Mechanical Turk to be useful as an infection vector, the cost to compromise a host must be less than the value derived from the host. Some percentage of workers who preview or accept the HIT will be vulnerable. In order to not arouse the suspicion of the workers, those who complete the HIT must be paid.

Table 2 answers the question: How much does it cost to compromise 1000 hosts if each completed HIT pays 5¢, accounting for the 10% Amazon fee? The second column shows the percentage of workers who accepted our HIT who had at least one plugin that was vulnerable per Section 4. The third, fourth, and fifth columns are

[4]Many of the plugins had multiple exploitable vulnerabilities for a particular version.

Table 2: Cost to compromise 1000 hosts at 5¢ per completed HIT.

	% vulnerable	% previewed	% accepted	% completed	cost ($/1000 hosts)
U.S.	84.9	99.5	87.9	81.0	52.52
India	96.3	99.5	87.6	80.2	45.83
Other	87.2	98.3	91.3	85.7	54.04

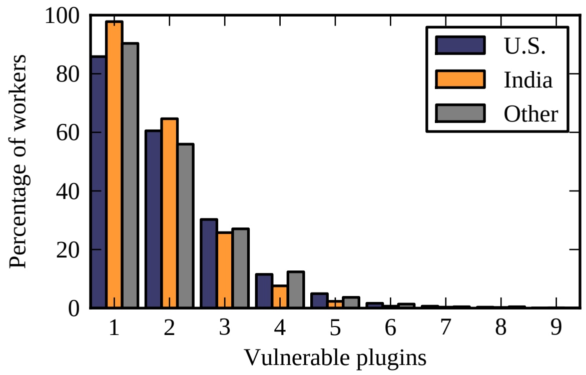


Figure 1: Percentage of workers who have at least n vulnerable plugins.

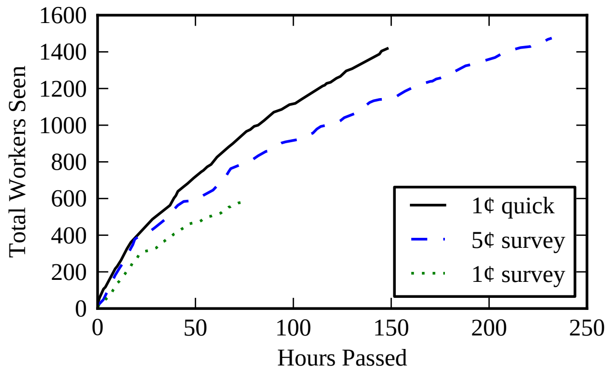


Figure 2: Cumulative number of workers seen per reward level.

the percentage of workers who, out of all those who interacted with the HIT, previewed, accepted, or completed the HIT, respectively. Some workers previewed the HIT without accepting and some accepted it without previewing, although this was far less common. Finally, the sixth column gives the cost in U.S. dollars to compromise 1000 workers' computers computed according to

$$cost = 1000 \times \$0.05 \times 110\% \times \frac{\%\ completed}{\%\ vulnerable} \qquad (1)$$

where the 110% accounts for Amazon's cut.

Figure 1 shows the percentage of workers who have at least n vulnerable plugins for each n. This shows that the majority of workers have at least two vulnerable plugins and a significant fraction have at least four.

Whereas the Panopticlick methodology can differentiate between different browser installs, we make the conservative assumption that at most one host can be compromised per IP address. Therefore, we only consider the first worker for a given IP address and HIT.

5.2 Infection rate

Infection rate is the counterpart to infection cost in an effective machine compromise strategy. In 2007, there were 100,000 workers [22]; two years later there were 400,000 workers [20]. While in the long term new installs will be limited by the number of newcomers to the Mechanical Turk worker pool, the initial infection rate will be constrained only by the total active workers, rate of interest in our HIT, and vulnerability rate.

The cumulative number of workers who accepted our HIT over time can be seen in Figures 2 and 3. Figure 2 illustrates the difference in rate controlling for the offered reward and task type, whereas Figure 3 focuses on geographic differences in rate within the 1¢ reward quick HIT.

The acceptance rate begins to wane at approximately 24 hours, with a generally flat rate beyond the first day. Table 3 expounds upon this dichotomy, presenting the first day rate and the long term rate over the rest of the data in our experiments. While the observed rates do not lend credence to using Mechanical Turk as the main infection vector for selling malware installs, this strategy can still be an ongoing supplement to other vectors.

5.3 Antivirus and program execution statistics

Since we ask the workers to run a program to determine if they have antivirus software installed and, if so, are the virus definitions up to date, we are able to collect statistics on both how many workers have no AV or out of date definitions and how many are willing to run programs for extra money. Further, we can measure how the size of the bonus affected the decision to run the program, at least for workers in India which provided the most information.

Table 4 aggregates the results of the bonus program—the third source of information described in Section 3.1. Of all workers who completed our HIT, 38% elected to participate in the bonus portion. We determined the uptake rate by instructing the workers to copy and paste result text into the Mechanical Turk form. A rather surprising number of workers using Windows have an an-

Table 3: Observed new worker rate expressed as hosts per day.

	1¢ survey		5¢ survey		1¢ quick	
	first day	later days	first day	later days	first day	later days
U.S.	58	31	61	29	176	90
India	204	77	291	73	197	61
Other	50	21	55	19	73	35

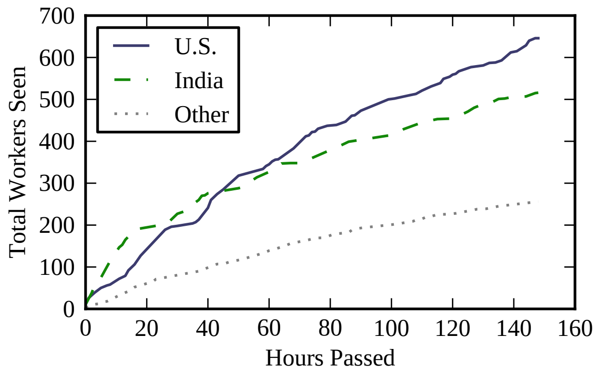


Figure 3: Cumulative number of workers seen per IP geolocation (1¢ reward quick HIT).

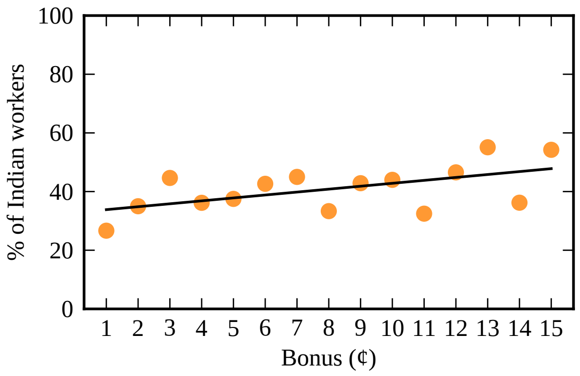


Figure 4: Percentage of Indian workers who ran the JScript for a given bonus.

Table 4: Percentage of Windows users with up to date AV

	AV installed (%)	up to date (%)
U.S.	98.7	22.8
India	92.7	68.7
Other	95.2	37.3

tivirus product installed. We attribute this to two factors. First, many OEMs ship Windows with an antivirus installed. Second, the instructions for the bonus were not clear that the worker would still receive the bonus if they ran the program even without any AV installed. We hesitate to rely on the self-reported numbers, however roughly 10% of workers on Windows responded that they had no AV installed. Further, there is likely to be a selection bias in which worker decided to do the HIT. The name of the HIT, “Anti-Virus Survey,” seems likely to deter workers who did not have antivirus installed. Therefore, the results in Table 4 are likely to overestimate the number of Mechanical Turk workers who have antivirus installed.

Since the size of the bonus offered to run the program was chosen uniformly at random between 1¢ and 15¢, we can compare the effect of price on the percentage of workers using Windows who opted to run the program. There are too few data points in each bin for U.S. and Other to draw any conclusions; however, for India, there is a clear trend. This is given in Figure 4. There is a $\rho = .56$ Pearson linear correlation between bonus and running the program with a p-value of $p = .029$.

6 Discussion

Our best information regarding how much client installs of malware cost comes from Caballero et at. [7] who report that 1000 unique installs on U.S. hosts cost between \$100 and \$180 and as little as \$7 or \$8 on Asian hosts. Even though we do not know how much a PPI service pays an affiliate for hosts, we assume that it is a similar order of magnitude, for example, half what the clients are charged. Table 2 shows that it costs roughly \$50/1000 hosts, both in the U.S. and around the world. For hosts in the U.S., Mechanical Turk is almost certainly an economically viable option using our parameters.

For Indian hosts, the situation is more complicated. As is, an affiliate could not make money selling hosts compromised with our set up; however, there are many factors that could influence the cost/benefit ratio. The most obvious change is that instead of paying 5¢, the HIT could pay 1¢. We do not have complete data for this case, but we believe that the percentage of vulnerable hosts is unlikely to change much compared to workers who would do a 5¢ HIT. Similarly, it seems unlikely that the percentage of workers who look at the preview (and thus could be compromised if they are vulnerable) but decide not to accept the HIT would go down. Thus the *% completed* in (1) is unlikely to increase. Therefore, we expect that simply changing from 5¢ to 1¢ would cause the cost of 1000 hosts to be cut by a fifth down to roughly \$10/1000 hosts. The biggest impact is on the rate at which workers can be compromised as can be seen in Figures 2 and 3.

At \$10/1000 hosts, the cost is slightly above the cost to clients, which would seem to imply that Mechanical Turk is too expensive a vector for lower-value hosts. There are a number of options yet unexplored for decreasing the cost or increasing the benefit. Many PPI affiliates sell the same compromised host to multiple PPI services [7]. If a host can be sold two or three times, then the benefit increases enough to justify the cost of compromise.

In addition to increasing benefit, one can drive down costs. Recall that the cost is proportional to the percentage of people who completed the HIT, among all of those who either viewed the preview page or accepted the HIT. One way to reduce the completed task would be to simply not allow (some percentage of) workers to submit or to refuse to pay them afterward. This is unlikely to be a viable solution in the long term as the workers will complain to Amazon and warn other workers on Mechanical Turk-oriented forums such as Turker Nation.[5] A better way to reduce the number who accept the HIT is to provide a vague description of the task and then on the preview page show a significantly more complicated task in the hopes that the workers will move on to easier ways to earn one penny. Alternatively, one could make the preview page appear non-functional with an error message or broken images to dissuade workers from accepting the HIT. Rather than risk not being able to complete the task, the worker will likely choose another HIT to accept.

6.1 Drawbacks

While the statistics on the vulnerability of workers' browsers to known exploits are striking, there are several possible sources of error in our estimate. Host-based or network-based antivirus could detect malicious code and lower the conversion rate or raise an alarm which causes workers to report the job to Amazon as malicious. The hosts could already be compromised and not viable Pay-Per-Install candidates, however we were not able to measure this possibility.

Perhaps the most glaring potential drawback is the scale at which this attack can be performed — some PPI affiliate programs expect to be provided with hundreds if not thousands of new installs on a daily basis and the Mechanical Turk worker pool can not sustain such a rate. We envision Mechanical Turk exploitation to be a supplementary infection stream, not necessarily an affiliate's sole vector.

Notwithstanding the low rate of infection, it is also fully possible that the startup cost of this exploitation vector would outweigh the marginal profits per infected host, even over long timescales. Additional data on the long-term dynamics of the Mechanical Turk workforce are needed to determine the asymptotic rate of new host acquisition.

Finally, this attack relies upon stealth and to a certain extent novelty to stay operational; discovery of new exploits and crafting of believable HITs would be necessary on an ongoing basis to maintain profitability. Even in the face of being detected and shut down by Amazon, however, it is not clear that creating new identities to act as requesters on Mechanical Turk would pose an insurmountable challenge to a motivated attacker.

6.2 Other capabilities

Although we have only studied the use of Mechanical Turk to compromise machines in the context of PPI, it is worth noting that this is but one instance of surreptitiously using a worker's machine to perform a task in exchange for a small monetary reward. Here we discuss several other possible improvements and attack vectors which might warrant future study.

A first optimization is that Mechanical Turk allows HITs to be targeted geographically. In this way, one could completely avoid paying workers in low-value countries, or tailor specific job costs to individual regions to maximize task completion while minimizing monetary outlay on a per-country basis.

One guarantee can be made about the machine being compromised through this vector: that its user has a worker account on Mechanical Turk. This tautological observation means that rather than selling installs in the PPI market (or perhaps in addition to doing so), the worker's Mechanical Turk account could be looted, transferring the worker's earned rewards to the attacker's bank. For workers who derive a significant fraction of their income from Mechanical Turk — which could be as high as 50% of the workers [16] — this could be quite lucrative. It is also much higher risk since workers are likely to notice the missing money and complain to Amazon.

Additionally, a malicious requester could also exploit workers via a clickjacking attack, to force them to unknowingly click on hidden ads or Facebook "Like" buttons [27]. While Mechanical Turk has in the past been rife with questionable or fraudulent HITs, the requester would be explicitly instructing the worker to perform whatever questionable task was desired [17]. In this format, a non-questionable HIT would be served in order to entice the user to fall victim to these attacks.

6.3 Possible solutions

At a fundamental level, the Mechanical Turk host compromise strategy relies upon the imbalance between the selling price of a compromised host and the price of purchasing access to a vulnerable machine. As our results show, the current state of affairs is such that this attack is economically viable under our set of assumptions. One

[5] `http://turkers.proboards.com`

can imagine this imbalance being fixed in several ways: Amazon could extend the capability of their Mechanical Turk interface such that the `iframe` interface is no longer needed or allowed; end-users could be educated about keeping antivirus and web plugins up to date; or browser developers could streamline the patching process. All of these defenses rely on lowering the supply of compromised hosts, when in fact it is the intersection of supply and *demand* that makes this compromise strategy viable.

Demand, in turn, can be lowered in many ways, including making "cash-out" of compromised machines more difficult through more stringent e-banking security, or likewise lowering the profitability of other underground enterprises such as pharmaceutical spam. Ultimately, an approach that shifts both supply of and demand for compromised end hosts should be pursued if the security community wishes to effectively squelch the Internet's underground economy.

6.4 Legality and ethics

We conclude this section by discussing the legal and ethical issues which arose during the course of this work.

Even though U.S. law regarding the enforceability of website terms of service is not settled, we were very careful to strictly adhere to the Amazon Mechanical Turk Participation Agreement [4] and general policies [3]. In particular, general policies forbid "HITs that require Workers to download software." Since all HITs by their very nature require downloading HTML and other code run in a web browser — even to view the preview — we interpret code to mean code run on the computer, outside of a browser context. To comply with this, we offer workers a small *bonus* if they will download and run our (benign) JScript code, but it is not required to complete the HIT. The JavaScript run in the browser for plugin detection is standard JavaScript used by, for example, the Electronic Frontier Foundation.

Since an action may be unethical and yet legal, we carefully considered what we were willing to do without the workers' knowledge, and for what we would ask for explicit consent. Although we do not explain the full technical process or ramifications, we do explicitly inform the user that we will be collecting non-personal information about their browser as part of our survey. Before accepting the HIT, the worker is informed that "the HIT page will also collect non-personal information about your web browser capabilities via JavaScript."

When visiting any website, browsers provide a great deal of information regarding the content they are willing and able to display to the JavaScript served by the website. In particular, when a worker see a preview of our HIT, a webpage is fetched and displayed in a frame and at this time, we could have collected most of the information of interest, even if the Worker then chose not to accept the HIT. Even though this would not violate the Participation Agreement, we fell this is unacceptable as we would be deriving a benefit from a worker without paying for it. Therefore, no information is collected from workers' browsers when they chose not to accept our HIT (other than the fact that the HIT is not accepted).

When offering the bonus to run a program to determine if the workers' antivirus software is up to date, there is no penalty for not opting in to this portion of the survey. Furthermore, we clearly describe the purpose of the program and suggest that the worker may inspect the program to verify the veracity of our claim — the program is an unobfuscated 58-line human readable text file. There is no deception in our description and after running the code, workers can choose not to paste the results back into the webpage.

Overall, we believe that our performance and disclosure during these experiments are on sound ethical footing because we (a) do not expose the user to any malicious code, (b) do not collect any personally identifiable information, and (c) do not collect *any* information until *after* the user has been informed of our intent — either by the HIT description or by the description of the bonus — and accepted the HIT or the offer of a bonus.

Finally, we note that this work does not require human subjects approval as per federal regulations[6] and our institution's guidelines.

7 Conclusions and future work

In this paper, we show that Mechanical Turk can profitably be used as a vector for compromising high-value computers for sale in the Pay-Per-Install market. The rate at which machines can be compromised is too low for Mechanical Turk to be a primary infection vector, but it could be used to supplement a PPI affiliate's primary source of compromised machines. As a bonus, we collected statistics on worker's antivirus software as well as their willingness to run arbitrary programs for a small reward; we also corroborated browser plugin vulnerability measurements taken by other researchers within a different population, suggesting their accuracy and general applicability [29].

At the end of the day, Mechanical Turk is only one simple way to drive traffic to our website. There are many different ways to buy traffic [30] that could prove even more effective in terms of cost per visit and quantity of vulnerable machines. We leave to future work the examination of the economic viability of purchasing other types of traffic for host compromise.

[6]"*Human subject* means a living individual about whom an investigator... conducting research obtains (1) Data through intervention or interaction with the individual, or (2) Identifiable private information," Protection of Human Subjects 45 C.F.R. §46.102(f) (2009).

Acknowledgments

We thank the anonymous reviewers for their helpful comments. This paper benefited from discussions with Stefan Savage, Hovav Shacham, and Geoffrey M. Voelker. This material is based upon work supported by the National Science Foundation under Grants No. CNS-0831532, CNS-0964702, CNS-0963702, NSF-0433668, and NSF-0831138; and by the Office of Naval Research under MURI Grant No. N000140911081.

References

[1] Mustafa Acer and Collin Jackson. Critical vulnerability in browser security metrics. In Collin Jackson, editor, *Proceedings of W2SP 2010*. IEEE Computer Society, May 2010.

[2] Gaurav Aggarwal, Elie Bursztein, Collin Jackson, and Dan Boneh. An analysis of private browsing modes in modern browsers. In Ian Goldberg, editor, *Proceedings of USENIX Security 2010*. USENIX, August 2010.

[3] Amazon Mechanical Turk General Policies. `https://www.mturk.com/mturk/help?helpPage=policies`.

[4] Amazon Mechanical Turk Participation Agreement, April 2009. `https://www.mturk.com/mturk/conditionsofuse`.

[5] David W. Baker, Steven M. Christey, William H. Hill, and David E. Mann. The development of a common enumeration of vulnerabilities and exposures. In Deborah Frincke and Ming-Yuh Huang, editors, *Proceedings of RAID 1999*. IBM BRS/SANS, September 1999.

[6] Adam Barth, Collin Jackson, and John C. Mitchell. Robust defenses for cross-site request forgery. In Paul Syverson and Somesh Jha, editors, *Proceedings of CCS 2008*. ACM, October 2008.

[7] Juan Caballero, Chris Grier, Christian Kreibich, and Vern Paxson. Measuring pay-per-install: The commoditization of malware distribution. In David Wagner, editor, *Proceedings of USENIX Security 2011*. USENIX, August 2011. To appear.

[8] Lydia B. Chilton, John J. Horton, Robert C Miller, and Shiri Azenkot. Task search in a human computation market. In Raman Chandrasekar et al., editor, *Proceedings of HCOMP 2010*, pages 1–9. ACM, July 2010.

[9] Nicolas Christin, Serge Egelman, Timothy Vidas, and Jens Grossklags. It's all about the Benjamins: An empirical study on incentivizing users to ignore security advice. In George Danezis, editor, *Proceedings of FC 2011*. IFCA, February 2011.

[10] Jia Deng, Wei Dong, Richard Socher, Li jia Li, Kai Li, and Li Fei-fei. Imagenet: A large-scale hierarchical image database. In Irfan Essa, Sing Bing Kang, and Marc Pollefeys, editors, *Proceedings of CVPR 2009*. IEEE Computer Society, June 2009.

[11] Peter Eckersley. How unique is your browser? In Mikhail Atallah and Nick Hopper, editors, *Proceedings of PETS 2010*, pages 1–18. Springer, July 2010.

[12] Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. Describing objects by their attributes. In Irfan Essa, Sing Bing Kang, and Marc Pollefeys, editors, *Proceedings of CVPR 2009*. IEEE Computer Society, June 2009.

[13] Jason Franklin, Vern Paxson, Adrian Perrig, and Stefan Savage. An inquiry into the nature and causes of the wealth of internet miscreants. In *Proceedings of CCS 2007*, pages 375–388. ACM, October 2007.

[14] Eric Gerds. Browser plugin detection with PluginDetect, March 2011. `http://www.pinlady.net/PluginDetect/`.

[15] Lin-Shung Huang, Eric Y. Chen, Adam Barth, Eric Rescorla, and Collin Jackson. Talking to yourself for fun and profit. In Helen J. Wang, editor, *Proceedings of W2SP 2011*. IEEE Computer Society, May 2011.

[16] Panagiotis G. Ipeirotis. Demographics of mechanical turk. CeDER Working Papers CeDER-10-01, Stern School of Business, March 2010.

[17] Panos Ipeirotis. Mechanical turk: Now with 40.92% spam. `http://behind-the-enemy-lines.blogspot.com/2010/12/mechanical-turk-now-with-4092-spam.html`, December 2010.

[18] Collin Jackson, Adam Barth, Andrew Bortz, Weidong Shao, and Dan Boneh. Protecting browsers from DNS rebinding attacks. In Sabrina De Capitani di Vimercati and Paul Syverson, editors, *Proceedings of CCS 2007*. ACM, October 2007.

[19] Kimberly. Cpalead sneaks into Amazon Mechanical Turk marketplace, August 2010. Online: `http://stopmalvertising.com/spam-scams/cpalead-sneaks-into-amazon-mechanical-turk-marketplace.html`.

[20] Nicholas Kolakowski. Amazon.com advocates crowdsourcing, Mechanical Turk in NYC. *eWeek.com*, November 2009.

[21] Gabriele Paolacci, Jesse Chandler, and Panagiotis G. Ipeirotis. Running experiments on amazon mechanical turk. *Judgment and Decision Making*, 5(5), August 2010.

[22] Jason Pontin. Artificial intelligence, with help from the humans. *New York Times*, March 2007.

[23] Joel Ross, Lilly Irani, M. Six Silberman Andrew Zaldivar, and Bill Tomlinson. Who are the crowdworkers?: Shifting demographics in Mechanical Turk. In *Proceedings of CHI EA 2010*, pages 2863–2872. ACM, April 2010.

[24] Paul Ruvolo, Jacob Whitehill, and Javier R. Movel-

lan. Exploiting structure in crowdsourcing tasks via latent factor models. Technical Report TR2010.01, Machine Perception Laboratory, 2010.

[25] M. Six Silberman, Lilly Irani, and Joel Ross. Ethics and tactics of professional crowdwork. *XRDS*, 17 (2):39–43, December 2010.

[26] Kevin Stevens. The underground economy of the pay-per-install (PPI) business, September 2009. `http://www.secureworks.com/research/threats/ppi`.

[27] Paul Stone. Next generation clickjacking. Presented at BlackHat Europe 2010, April 2010.

[28] Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-UCSD birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[29] Gilbert Wondracek, Thorsten Holz, Christian Plazer, Engin Kirda, and Christopher Kruegel. Is the internet for porn? an insight into the ouline adult industry. In Tyler Moore, editor, *Proceedings of WEIS 2010*. ACM, June 2010.

[30] Qing Zhang, Thomas Ristenpart, Stefan Savage, and Geoffrey M. Voelker. Got traffic? An evaluation of click traffic providers. In Carlos Castillo et al., editor, *Proceedings of WebQuality 2011*. WICOW/AIRWeb, March 2011.