It's All About The Benjamins:
An empirical study on incentivizing users to ignore security advice

Nicolas Christin[a], Serge Egelman[b], Timothy Vidas[c], and Jens Grossklags[d]

[a]INI/CyLab, Carnegie Mellon University
[b]National Institute of Standards and Technology
[c]ECE/CyLab, Carnegie Mellon University
[d]IST, Pennsylvania State University
nicolasc@andrew.cmu.edu
serge.egelman@nist.gov
tvidas@ece.cmu.edu
jensg@ist.psu.edu

Abstract. We examine the cost for an attacker to pay users to execute arbitrary code—potentially malware. We asked users at home to download and run an executable we wrote without being told what it did and without any way of knowing it was harmless. Each week, we increased the payment amount. Our goal was to examine whether users would ignore common security advice—not to run untrusted executables—if there was a direct incentive, and how much this incentive would need to be. We observed that for payments as low as \$0.01, 22% of the people who viewed the task ultimately ran our executable. Once increased to \$1.00, this proportion increased to 43%. We show that as the price increased, more and more users who understood the risks ultimately ran the code. We conclude that users are generally unopposed to running programs of unknown provenance, so long as their incentives exceed their inconvenience.

Key words: Behavioral Economics, Online Crime, Human Experiments

1 Introduction

Since the early 2000s, Internet criminals have become increasingly financially motivated [22]. Of particular concern is the large number of "bots," that is, compromised end user machines that can be accessed and controlled remotely by these miscreants. Bots are a security concern, because they are federated in centrally-operated networks ("botnets") that can carry out various kinds of attacks (e.g., phishing site hosting, spam campaigns, distributed denial of service, etc.) on a large scale with relative ease. Botnet operators usually do not launch these attacks themselves, but instead rent out computing cycles on their bots to other miscreants.

Estimates on the total number of bots on the Internet range, for the year 2009, from 6 to 24 million unique hosts [4, 31]. Individual botnets can be comprised of hundreds of thousands of hosts. For instance, researchers managed to hijack the Torpig botnet for a period of ten days, and observed on the order of 180,000 new infections over that period, and 1.2 million unique IP addresses contacting the control servers [30].

The scale of the problem, however alarming it may seem, suggests that most Internet users whose computers have been turned into bots either do not seem to notice they have been compromised, or are unable or unwilling to take corrective measures, potentially because the cost of taking action outweighs the perceived benefits. There is evidence (see, e.g., [29]) of hosts infected by multiple pieces of malware, which indicates that a large number of users are not willing to undertake a complete reinstallation of their system until it ceases to be usable. In other words, many users seem to be content ignoring possible security compromises as long as the compromised state does not noticeably impact the performance of the machine.

Consequently, bots are likely to be unreliable. For instance, they may frequently crash due to multiple infections of poorly programmed malware. Their economic value to the miscreants using them is in turn fairly low; and indeed, advertised bot prices can be as low as \$0.03-\$0.04 per bot,[1] according to a recent Symantec report [31].

Overall, bot markets are an interesting economic environment: goods are seemingly of low quality, and treachery amongst sellers and buyers is likely rampant [14], as transaction participants are all engaging in illegal commerce. Yet, the large number of bots, and the absence of notable decrease in this number, seem to indicate that bot markets remain relatively thriving. This puzzle is even more complicated by the fact most surveyed Internet users profess a strong desire to have secure systems [5].

In this paper, we demonstrate that, far from being consistent with their stated preferences, in practice, users actually do not attach any significant economic value to the security of their systems. While ignorance could explain this state of affairs, we show that the reality is much worse, as some users readily turn a blind eye to questionable activities occurring on their systems, as long as they can themselves make a modest profit out of it.

We describe an experiment that we conducted using Amazon's Mechanical Turk, where we asked users to download and run our "Distributed Computing Client" for one hour, with little explanation as to what the software actually did,[2] in exchange for a nominal payment, ranging from \$0.01 to \$1.00. We conducted follow-up surveys with the users who ran the software to better understand the demographics of at-risk populations. We use the results of this experiment to answer the following questions:

- **How much do Internet users value access to their machines?** If users who are at risk of getting infected could be compensated for the infection, how much should the compensation be?
- **Is there a specific population at risk?** Do we observe significant demographic skew for certain types of populations in our study?
- **Are current mitigations effective at dissuading users from running arbitrary programs that they download from the Internet?** When software warns users about the dangers of running unknown programs, are they any less likely to continue running those programs?

[1] In this entire paper, all prices are given in US dollars. That is, the "\$" sign denotes US currency.

[2] As discussed in Section 3, in reality, the software merely collected running process information and other system statistics, ran down a countdown timer, and reported this information back to us; in our experimental condition, it also prompted users for administrative access to run.

Answering these questions is important because it allows us to quantify how risk-seeking people may be, which in turn may help optimize user education and other mitigation strategies. These answers also provide an idea on the upper bound of the value of a bot, and permit us to compare prices set in actual, realized transactions, to advertised prices [14]. Being able to dimension such prices helps us better understand which intervention strategies are likely to be successful; for instance, if a security mitigation costs users one hour of work, while they value access to their machine at a few cents, users are unlikely to adopt the security mitigation [13].

2 Related Work

There has recently been a considerable amount of academic research devoted to trying to dimension economic parameters in the realm of online crime. Pioneering works in that realm mostly attempted to measure prices of illegitimately acquired commodities advertised in underground online markets [8, 32]. Industry research is also very active in that field. For instance, Symantec and Verizon issue yearly reports that dimension key economic indicators of the vitality of online criminal activity (see, e.g., [6, 31]).

Most of this research, though, relies on passive measurements; advertising channels are monitored, but researchers rarely become party to an actual transaction. As such, the prices monitored and reported may be quite different from the actual sales prices [14]. A few recent papers try to obtain more precise measurements by taking over (or infiltrating) botnets and directly observing, or participating in the transactions [18, 30].

To complicate matters, the reported value of botnets also varies based on perspective. From a prosecution perspective, costs of crimes may be inflated to seek larger sentences. From a botnet operator perspective, risks associated with operation may warrant hosting in countries that have ambiguous or non-existent cybercrime laws. It is the botnet renter's perspective that we explore. A US congressional report estimated renting an entire botnet in 2005 cost $200-$300 per hour [34]. Renting a bot in 2006 reportedly cost $3-$6 [21]. In 2007 studies show an asking cost as high as $2-$10 [8], but the actual price paid have been as low as $0.25 [35]. Recently, prices have been discerned to cost $50 for several thousand bots for a 24-hour period [24].

Other work aims at uncovering relationships between different online crime actors, and calculating economic metrics (e.g., actors, revenue, market volume, etc.) from network measurements. For instance, Christin et al. [7] gathered four years worth of data on an online scam prevalent in Japan, and show that the top eight groups are responsible for more than half of the malicious activity, and that considerable profit (on the order of $100,000 per year) can be amassed for relatively little risk (comparatively small fines, and very light prison sentences). Likewise, Moore and Edelman [23] assessed the economic value of typosquatting domains by gathering a large number of them and evidence some disincentives for advertisement networks to intervene forcefully.

Closer to the research on which we report in this paper, Grossklags and Acquisti provide quantitative estimates of how much people value their privacy. They show that most people are likely to give up considerable private information in exchange for $0.25 [11]. Similarly, Good et al. conducted a laboratory study wherein they observed whether participants would install potentially risky software with little functional benefit [10].

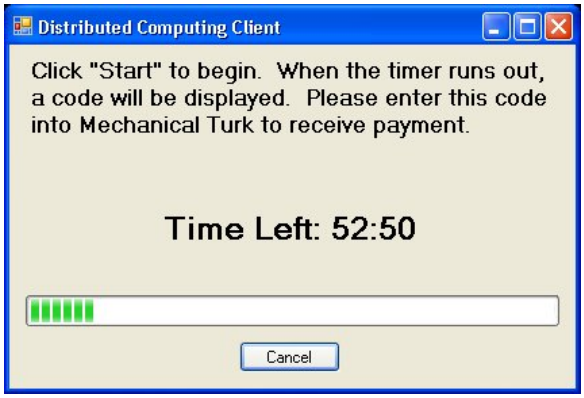

Fig. 1. Screenshot of the "Distributed Computing Client." After subjects clicked a start button, a timer ran for one hour. Once the timer expired, a completion code was displayed. Subjects entered this completion code into Mechanical Turk in order to receive payment.

Participants overwhelmingly decided to affirmatively install peer-to-peer and desktop-enhancing software bundled with spyware. Most individuals only showed regret about their actions when presented with much more explicit notice and consent documents in a debriefing session. To most participants, the value extracted from the software (access to free music or screen savers) trumped the potentially dangerous security compromises and privacy invasions they had facilitated.

We complement these works by trying to provide a precise measurement of how little it could cost an attacker to have unfettered access to a system with the implicit consent of the owner. We also contribute to a better understanding of behavioral inconsistencies between users' stated security preferences [5] and their observed actions.

3 Experimental Methodology

The key part of our study lies in our experiment, which we ran using Amazon's Mechanical Turk. The objective was to assess at what price, and under which conditions, would people install an unknown application. In this section, we describe the study environment and two different conditions we created. We analyze our results in Section 4.

Mechanical Turk is a marketplace for pairing "workers" with "requesters." Requesters post tasks that are simple for humans to complete, but difficult to automate (e.g., image tagging, audio transcription, etc.). In return, the requesters pay workers micropayments—on the order of a few cents—for successfully completing a task. However, recently researchers have begun using it as a crowd-sourcing tool for performing studies on human subjects [19]. This is because the demographics of Mechanical Turk do not significantly differ from the demographics of worldwide Internet users [27]. But what about the quality of the resulting data? To collect data for a 2008 study, Jakobsson posted a survey to Mechanical Turk while also hiring a market research firm. He concluded that the data from Mechanical Turk was of the same quality, if not higher [16].

In September of 2010, we created a Mechanical Turk task offering workers the opportunity to "get paid to do nothing." Only after accepting our task did participants see a detailed description: they would be participating in a research study on the "CMU Distributed Computing Project," a fictitious project that we created. As part of this, we instructed participants to download a program and run it for an hour (Figure 1). We did not say what the application did. After an hour elapsed, the program displayed a code, which participants could submit to Mechanical Turk in order to claim their payment.

Because this study involved human subjects, we required Institutional Review Board (IRB) approval. We could have received a waiver of consent so that we would not be required to inform participants that they were participating in a research study. However, we were curious if—due to the pervasiveness of research tasks on Mechanical Turk—telling participants that this was indeed a research task would be an effective recruitment strategy. Thus, all participants were required to click through a consent form. Beyond the consent form, there was no evidence that they were participating in a research study; all data collection and downloads came from a third-party privately-registered domain, and the task was posted from a personal Mechanical Turk account not linked to an institutional address. No mention of the "CMU Distributed Computing Project" appeared on any CMU websites. Thus, it was completely possible that an adversary had posted a task to trick users into downloading malware under the guise of participating in a research study, using a generic consent form and fictitious project names in furtherance of the ruse.

We reposted the task to Mechanical Turk every week for five weeks. We increased the price each subsequent week to examine how our results changed based on the offering price. Thus, the first week we paid participants $0.01, the second week $0.05, the third week $0.10, the fourth week $0.50, and the fifth week $1.00. In order to preserve data independence and the between-subjects nature of our experiment, we informed participants that they could not participate more than once. We enforced this by rejecting results from participants who had already been compensated in a previous week.

We further restricted our task to users of Microsoft Windows XP or later (i.e., XP, Vista, or 7).[3] In Windows Vista and later, a specific security mitigation is included to dissuade users from executing programs that require administrator-level access: User Account Control (UAC). When a program is executed that requires such access, a prompt is displayed to inform the user that such access is requested, and asks the user if such access should be allowed (Figure 2). To examine the effectiveness of this warning, we created a second between-subjects condition. For each download, there was a 50% chance that the participant would download a version of our software that requested administrator-level access via the application manifest, which in turn would cause the UAC warning to be displayed prior to execution.

The main purpose of this experiment was to collect data on the ratio of people who downloaded the application versus viewed the task, and the ratio of people who ran the application versus downloaded the application, as well as how these ratios changed

[3] Certain commercial software is identified in this paper in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the software identified is necessarily the best available for the purpose.

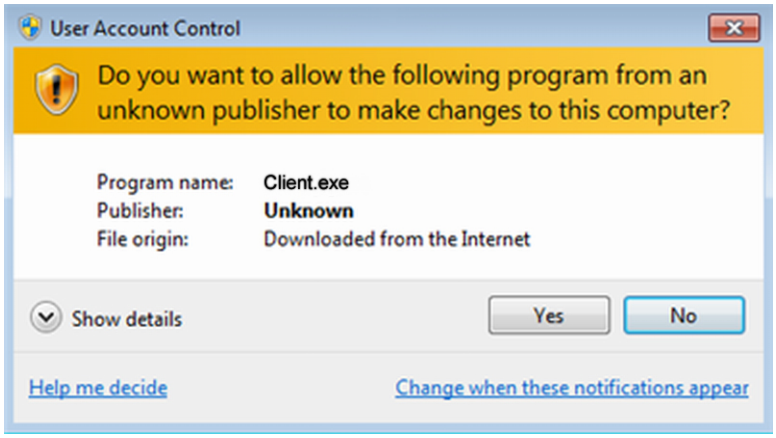

Fig. 2. UAC prompt. Users of Windows Vista or 7 saw this prompt in the experimental condition.

based on the payment amount and presence of the UAC warning. Finally, we collected various (anonymous) data from the systems of the participants who ran the application. We designed our application to report anonymous statistics using port 123, the port normally reserved for the Network Time Protocol, since we reasoned that outgoing connections on this port were least likely to be firewalled (beyond the HTTP ports, which were already in use on our server). We specifically collected the following information:

- **Experimental Condition**— The application reported its version so we could examine if participants were disproportionally executing the application in either the experimental (UAC) or control condition.
- **Version**— We collected Windows versions to examine whether participants were exposed to the UAC prompt (i.e., XP users who executed the application in the experimental condition were not exposed to the prompt because the UAC feature did not exist in their version). Additionally, this also gave us insights into how up-to-date their software was (i.e., whether they had the latest service packs installed).
- **Process List**— We collected a list of the image names for all running processes to examine whether participants were taking any security precautions. Specifically, we were looking for known anti-virus (AV) software, as well as indications that they were running the software from within a virtual machine (VM).
- **Virtual Machine Detection**— To further detect whether our application was being executed from within a virtual machine (VM), we performed two heuristics. First, we used the Red Pill VM-detection routine to report the memory address of the Interrupt Descriptor Table (IDT) and number of CPUs [28]. Second, we reported the name of the motherboard manufacturer. Some VM software will report a generic manufacturer rather than forwarding this information from the host operating system. Obviously this is an incomplete list of VM-detection techniques, but we believe these heuristics allowed us to show a lower bound on VM usage.

	$0.01	**$0.05**	**$0.10**	**$0.50**	**$1.00**
viewed task	291	272	363	823	1,105
downloaded	141 49%	135 50%	190 52%	510 62%	738 67%
ran	64 45%	60 44%	73 38%	294 58%	474 64%

Table 1. Experimental results showing the Mechanical Turk users who viewed the task; of those, the fraction who downloaded the program; and of those, the fraction who ran the program. The latter represents only those who submitted data to our server, and therefore is a lower bound.

Other than collecting this information, our program did absolutely nothing other than running a timer and periodically reporting to our server that it was still running.

After participants submitted valid payment codes at the end of the experiment, we invited them to complete an exit survey for a $0.50 bonus payment. This bonus payment was identical across all experimental conditions. This survey was designed to answer questions about the participants' risk perceptions during and outside of the study, what security precautions they normally take, and various demographic information.

4 Results

During the course of our five-week study, our task was viewed 2,854 times. This corresponds to 1,714 downloads, and 965 confirmed executions. We found that the proportion of participants who executed the program significantly increased with price, though even for a payment of $0.01, 22% of the people who viewed the task downloaded and executed the program (Table 1). This raises questions about the effectiveness of well-known security advice when competing against the smallest of incentives. In this section we analyze our results with regard to differing behaviors among the pricing conditions, data collected from participants' systems, and the exit survey data.

4.1 Price Points

Overall, we observed statistically significant differences between the payment amounts with regard to the fraction of participants who downloaded the program after viewing the task description ($\chi^2_4 = 59.781$, $p < 0.0005$). Upon performing post-hoc analysis using Fisher's exact test, we observed that this was due to differences between the $0.50 and $1.00 price points and each of the three lowest price points ($p < 0.002$ and $p < 0.0005$, respectively). We also observed that significantly more people downloaded the program when the price was raised from $0.50 to $1.00 ($p < 0.030$).

Once participants downloaded the program, our only indication that they had executed it was when our server received data from the program. If a firewall prevented users from reaching port 123 on our server, we assumed that they had not run the program. Thus, our reported proportion of executes-to-downloads represents a lower bound. Nevertheless, we observed statistically significant differences between the conditions ($\chi^2_4 = 58.448$, $p < 0.0005$). As was the case with the proportion of downloads, post-hoc analysis showed that this was due to the increase in execution at the $0.50

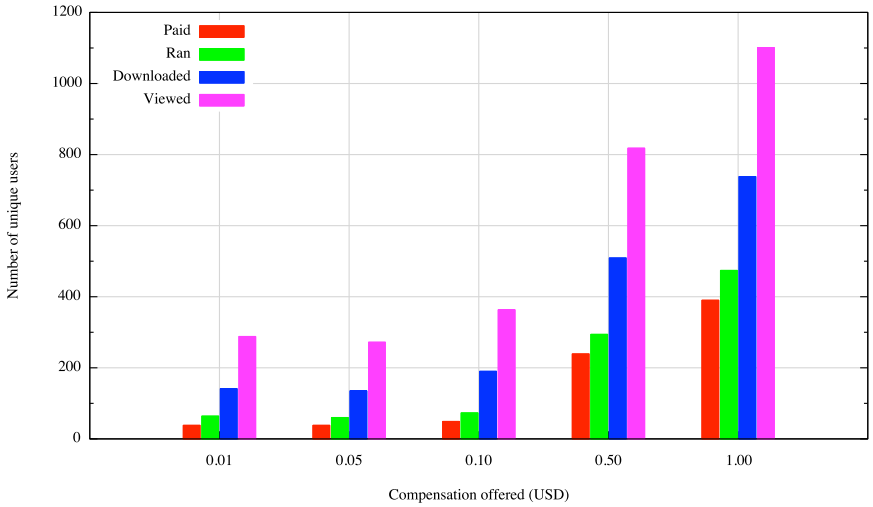

Fig. 3. Task enrollment and completion rates. "Paid" corresponds to users who ran the program for a full hour and correctly submitted a payment code; "Ran" corresponds to users who ran the program, no matter how briefly; "Downloaded" and "Viewed" correspond to users who downloaded our program and viewed the task, respectively. We noticed statistically significant differences in participants' behaviors at the \$0.50 and \$1.00 price points.

and \$1.00 price points ($p < 0.01$ and $p < 0.0001$, respectively). Likewise, significantly more people ran the program when the price was raised from \$0.50 to \$1.00 ($p < 0.021$).

Thus, participants' behaviors did not change relative to the price we were paying them until it was increased to \$0.50, at which point not only did we see a dramatic increase in the total number of people viewing the task, but also a significant increase in the proportion who opted to perform the task. The increase in the former is purely an artifact of our experimental platform, Mechanical Turk, indicating that many users are unwilling to view tasks that pay less than \$0.50. The latter increase is more important since we observed differences in participants' behaviors as a function of price. These changes in participants' behaviors continued as the price was further increased to \$1.00.

4.2 Participant Behavior

When we compared the participants who were exposed to the UAC dialog (Figure 2) with the participants in the control condition, we found no significant differences.[4] Thus, regardless of price, the warning had no observable effect on participants' willingness to download and execute an unknown program—even one that required them

[4] It is possible that some participants in the UAC condition had disabled this feature, which we did not attempt to detect. However, we do not believe these participants confound our findings since they provide a realistic snapshot of UAC usage.

to explicitly grant it administrator-level permissions. Because of the lack of effect, we disregarded this variable in our analysis.

Because the Red Pill VM-detection routine [28] only works reliably on single-CPU computers, we also collected information on the number of CPUs. Using Red Pill, we detected the presence of a VM on a single participant's machine. Examining each participants' process lists, we were able to confirm that this participant was running VMware. Additionally, we detected VMware Tools running on an additional fifteen machines (sixteen in total), and Parallels Tools running on a single machine. Thus, we can confirm that at least seventeen participants (1.8% of 965) took the precaution of using a VM to execute our code.[5] Eleven of these participants were in the $1.00 condition, five were in the $0.50 condition, and one was in the $0.01 condition. The information we collected on participants' motherboards was not useful in determining VM usage.

When the program was executed, we received a list of the other processes that each participant was running. In this manner we amassed a list of 3,110 unique process names, which we categorized as either corresponding to malware or security software using several online databases. Because we only recorded the image names and not the sizes, file locations, or checksums, these classifications are very rough estimates. We took great care to only classify something as malware after being completely certain; this created a lower bound on the number of infected systems. Conversely, we took a liberal approach to categorizing something as security software in order to create an upper bound; it is possible—indeed likely—that some of the image names masquerading as security software were actually malware. All in all, we found no significant differences between the pricing conditions with regard to malware infections or the use of security software; at least 16.4% (158 of the 965 who ran the program) had a malware infection, whereas as many as 79.4% had security software running (766 of 965). Surprisingly, we noticed a significant positive trend between malware infections and security software usage ($\phi = 0.066$, $p < 0.039$). That is, participants with security software were *more* likely to also have malware infections (17.6% of 766), whereas those without security software were *less* likely to have malware infections (11.6% of 199). While counterintuitive, this may indicate that users tend to exhibit risky behavior when they have security software installed, because they blindly trust the software to fully protect them.

Once we increased the payment to $0.50, participants made different decisions with regard to their system security. We collected data on their Windows versions in order to examine if they had applied the most recent service packs. We considered this a proxy for risk aversion—similar to the use of security software; in the computer security context, higher risk aversion should correspond to an increase in up-to-date systems. Indeed, we observed this effect: 72.1% and 67.5% of participants who were paid $0.50 and $1.00, respectively, had a current version of Windows, compared to an average of 54.3% of participants who were paid less ($p < 0.0005$ for Fisher's exact test).

Finally, we found that the better-compensated participants also performed the task more diligently. We examined the number of participants in each condition who submitted an invalid payment code (Table 2) and found that "cheating" significantly decreased as the payment increased ($\chi^2_4 = 52.594$, $p < 0.0005$). Likewise, we found a statistically

[5] We do not know if this was a precaution or not. It is equally likely that participants were simply using VMs because they were Mac or Linux users who wanted to complete a Windows task.

	$0.01	**$0.05**	**$0.10**	**$0.50**	**$1.00**
average runtime in minutes	39.30	43.67	43.63	52.02	52.65
percentage running the program for 60 minutes	59.4%	63.3%	67.1%	81.3%	82.3%
percentage of invalid codes submitted	46.5%	28.3%	24.6%	12.8%	14.8%

Table 2. For each payment amount, each row depicts: the average number of minutes participants spent running the program (out of 60), the percentage of participants who ran the program for the full 60 minutes, and the percentage of participants who attempted to cheat by submitting an invalid payment code (out of the total number of submitted payment codes).

significant correlation between time spent running the executable and payment amount ($r = 0.210$, $p < 0.0005$).

Thus, we observed that not only did the $0.50 and $1.00 price points allow us to recruit more participants, but the overall quality—reliability and modernity—of the machines recruited and the diligence of their operators considerably increased as well.

The measurements we obtained through instrumentation of our web site and of our software give us a pretty detailed picture of user behavior when facing decisions about whether to run untrusted code or not. We were curious to further examine the "at-risk" population of users who elected to run untrusted code, and thus completed our experiment, in terms of demographic and behavioral characteristics. To that effect, after completing the experimental portion of this study, we invited participants who had submitted a valid payment code to complete an exit survey for a bonus payment of $0.50.

4.3 Self-Reported Data

In the exit survey we collected information about participants' demographics, their risk perceptions, and their stated security practices.

Demographics. A total of 513 participants completed the survey. Forty percent of respondents came from India, 30% from the US or Canada, and the majority of the rest were from Europe.[6] Using the United Nations' list of developed countries [33], we partitioned participants based on their nationalities. We observed that the proportion of participants from the developed world increased with the payment amount: from 9.4% to 23.4%. We found that this increase was statistically significant when comparing the $0.01 condition to the $0.50 and $1.00 conditions ($p < 0.02$ and $p < 0.01$, respectively), but was not significant when comparing the other conditions. We did not observe significant differences with regard to age (mean $\mu = 28.1$ years old, with standard deviation $\sigma = 8.95$ years) or gender (349 males and 164 females).

We gauged participants' security expertise with four questions:

- Do you know any programming languages?
- Is computer security one of your main job duties?

[6] Thirty-three participants listed their nationalities as "Caucasian." After unsuccessfully trying to locate Caucasia on a map, we omitted their responses to this question.

– Have you attended a computer security conference or class in the past year?
– Do you have a degree in computer science or another technical field (e.g. electrical engineering, computer engineering, etc.)?

We observed no differences between the conditions with regard to the individual conditions nor when we created a "security expert score" based on the combination of these factors. That is, the proportion of people who answered affirmatively to each of the questions remained constant across the price intervals. So why did participants execute the program?

Security concerns. Windows' automatic update functionality is disabled when it determines that the software may have been pirated. We observed a significant correlation between participants from the developing world and unpatched versions of Windows ($\phi = 0.241$, $p < 0.0005$), which hints at a correlation between software piracy and the developing world. However, we do not believe that this effect is solely responsible for the changes in behavior as a function of price that we observed during our study.

We asked participants to rate the danger of running programs downloaded from Mechanical Turk using a 5-point Likert scale. What we found was that as the price went up, participants' perceptions of the danger also increased ($F_{508,4} = 3.165$, $p < 0.014$). Upon performing post-hoc analysis, we observed that risk perceptions were similar between the \$0.01, \$0.05, and \$0.10 price points ($\mu = 2.43$, $\sigma = 1.226$), as well as between the \$0.50 and \$1.00 price points ($\mu = 2.92$, $\mu = 1.293$); once the price hit \$0.50 and above, participants had significantly higher risk perceptions ($t_{511} = 3.378$, $p < 0.001$). This indicates that as the payment was increased, people who "should have known better" were enticed by the payment; individuals with higher risk perceptions and awareness of good security practices did not take the risk when the payment was set at the lower points, but ignored their concerns and ran the program once the payment was increased to \$0.50. These perceptions were not linked to demographic data.

These results suggest that participants may have felt an increased sense of security based on the context in which our program was presented. As mentioned earlier, though, there was no actual proof this was a legitimate university study, as we purposefully stored the executable on a third-party server and did not provide any contact information.

We periodically browsed Mechanical Turk user forums such as Turker Nation [2] and Turkopticon [3], and noticed with interest that our task was ranked as legitimate because we were paying users on time and anti-virus software was not complaining about our program. Clearly, neither of these statements is correlated in any way to potential security breaches caused by our software. Anti-virus software generally relies on malware being added to blacklists and may not complain about applications that were given full administrative access by a consenting user. Perhaps a warning to the effect that the application is sending network traffic would pop up, but most users would dismiss it as it is consistent with the purpose of a "Distributed Computing Client." As such, it appears that people are reinforcing their false sense of security with justifications that are not relevant to the problem at hand.

Last, one Turkopticon user also had an interesting comment. He incorrectly believed we were "running a port-scanner," and was happy with providing us with potentially

private information in exchange for the payment he received. This echoes Grossklags and Acquisti's finding [11] that people do not value privacy significantly.

5 Discussion and Conclusions

In this experiment, we observed what economists have known for centuries: it really is all about the Benjamins.[7] We show in the security context how users can be motivated by direct incentive payments to ignore commonly known security advice. Even though around 70% of all our survey participants understood that it was dangerous to run unknown programs downloaded from the Internet, all of them chose to do so once we paid them. We also demonstrate that the strength of user participation is consistent with basic economic principles, so that more money equals more interest in the task, downloads and executions of potentially harmful code, and commitment to the instructions given [9, 15]. This study serves as a clarion call to security researchers and practitioners: security guidance should not only create incentives for compliance with advice and policies, but also explicitly include mechanisms to support users who are prone to trade away security cheaply.

When investigating the correspondence between security concerns and security behaviors more closely, we found that subjects in the high payment condition were more concerned about the risks of downloading programs while doing online work. Similarly, these participants were on average better suited to withstand security threats (i.e., they owned more recently patched machines). In other words, more security-conscious and security-aware subjects were lured into our project with the higher available remuneration. On the one hand, these subjects act consistently in applying security patches to react to their heightened concerns. On the other hand, they waste all their hard security efforts by consensually inviting undocumented code on their machines and bypassing Windows security stopping blocks (i.e., UAC).

There are a number of interpretations for our observations that fit the model of the *bounded rational* computer user. In particular, participation may be due to a strong pull of "immediate gratification" and revealing of time-inconsistent preferences [5]. Further, the task triggers "relative thinking." That is, we have to consider the promised ease of our task in conjunction with a wage that is appealing relative to many more cumbersome Mechanical Turk offerings, e.g., annotating pictures or transcribing recordings [17]. As a result, individuals further neglect to consider the harm that they may inflict on themselves. Relative thinking also helps to account for the surprising diligence of our subjects in the $0.50 and $1.00 conditions. At the same time, even in the $0.01 condition, 22% of the users who viewed the task downloaded and ran the program. This indicates that even for a negligible remuneration, Internet users are willing to bear significant risk; indeed a better title for this paper may be "It's All About The Abrahams.[8]"

We further believe that a heightened sense of "false security" explains the involvement of many subjects. First, many of the more concerned individuals owned better patched machines and therefore may have erroneously believed that seemingly legitimate tasks were unlikely to cause harm. Second, this observation was mirrored by the

[7] Benjamin Franklin is pictured on US $100 bills, hence the slang "Benjamin" for banknotes.
[8] Abraham Lincoln is pictured on US $0.01 coins.

entire user population. We found that the presence of security software was correlated with the number of infections with malicious code on user machines. Third, users' discussions of our task on message boards may have enticed others to participate despite the lack of any security assurances and tests. Moreover, such messages could have easily been posted by co-conspirators.

To explain why *rational* individuals who are concerned about security and who undertake precautions (i.e., install patches and anti-virus software) nevertheless fail to act securely, we can draw from the safety regulation literature. Most prominently, Peltzman outlined in his theory of risk compensation that individuals evaluate the combined demand for safety and usage intensity (e.g., faster driving on a highway) [25]. In the online security context this means that users with more security-ready systems—irrespective of whether they safeguard themselves or administrators act for them—are comparatively more likely to engage in risky behaviors. For example, they may be more likely to seek free adult content or download undocumented code/media that increases their perceived utility from experience. While this process may be driven by selfish and rational considerations for at least some users, it always increases the risk of collateral damage due to negative externalities.

In the online security context, this behavior is particularly significant because malware developers not only carefully conceal their code from users, but also limit the adverse impact on hosts' systems. Distributed attacks may be implicitly tolerated by users as long as they are not directly affected, and lead to what Bill Cheswick likens to a "toxic waste dump" billowing blue smoke across the Internet [26]. Miscreants can benefit from these negative externalities by harvesting resources at a price far below the social cost they are inflicting on the user population.

In conclusion, providing direct incentives for the installation of undocumented code constitutes another malware distribution channel. In this manner, the notion of a "Fair Trade" botnet is not out of the realm of possibility.[9] The occurrence of such an approach may be relatively short-lived such as in previous examples with some free centralized systems (e.g., Download.com [20]), if distributed work platforms exercise considerably more control over the advertised tasks. However, a payment scheme could also give rise to unforeseen semi-legitimate business models that cannot be easily outlawed.

We argue that once payment is high enough and inconvenience is low enough, users have little incentive to comply with security requests [13]. Worse, behavioral biases further magnify the impact of negative externalities in network security [12]. By allowing attackers to use their machines for further attacks, users themselves become a threat to the network. We hope this research will stimulate further discussions on how to realign user incentives, that have seemingly drifted very far away from what would be needed to produce a desirable outcome.

[9] "Fair Trade" is a certification for foods coming from producers who receive a living wage [1].

Acknowledgments

This paper greatly benefited from discussions with Stuart Schechter, David Molnar, and Alessandro Acquisti. Lorrie Cranor assisted with IRB documentation. This work is supported in part by CyLab at Carnegie Mellon under grant DAAD19-02-1-0389 from the Army Research Office, and by the National Science Foundation under ITR award CCF-0424422 (Team for Research in Ubiquitous Secure Technology) and IGERT award DGE-0903659 (Usable Privacy and Security).

References

1. Fair Trade USA. `http://www.transfairusa.org/`.
2. Turker nation. `http://www.turkernation.com`.
3. Turkopticon. `http://turkopticon.differenceengines.com`.
4. B. Acohido. Are there 6.8 million – or 24 million – botted PCs on the Internet? `http://lastwatchdog.com/6-8-million-24-million-botted-pcs-internet/`. Last accessed September 16, 2010.
5. A. Acquisti and J. Grossklags. Privacy and rationality in individual decision making. *IEEE Security & Privacy*, 3(1):26–33, January–February 2005.
6. W. Baker, A. Hutton, C. Hylender, C. Novak, C. Porter, B. Sartin, P. Tippett, and J. Valentine. 2009 Data breach investigations report. *Verizon Business Security Solutions*, Apr. 2009.
7. N. Christin, S. Yanagihara, and K. Kamataki. Dissecting one click frauds. In *Proceedings of the Conference on Computer and Communications Security (CCS)*, pages 15–26, Chicago, IL, Oct. 2010.
8. J. Franklin, V. Paxson, A. Perrig, and S. Savage. An inquiry into the nature and causes of the wealth of internet miscreants. In *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, pages 375–388, Alexandria, VA, October 2007.
9. S. Gaechter and E. Fehr. Fairness in the labour market - A survey of experimental results. In F. Bolle and M. Lehmann-Waffenschmidt, editors, *Surveys in Experimental Economics, Bargaining, Cooperation and Election Stock Markets*. Physica Verlag, 2001.
10. N. Good, R. Dhamija, J. Grossklags, S. Aronovitz, D. Thaw, D. Mulligan, and J. Konstan. Stopping spyware at the gate: A user study of privacy, notice and spyware. In *Proceedings of the Symposium On Usable Privacy and Security (SOUPS 2005)*, pages 43–52, Pittsburgh, PA, July 2005.
11. J. Grossklags and A. Acquisti. When 25 cents is too much: An experiment on willingness-to-sell and willingness-to-protect personal information. In *Proceedings (online) of the Sixth Workshop on Economics of Information Security (WEIS)*, Pittsburgh, PA, 2007.
12. J. Grossklags, N. Christin, and J. Chuang. Secure or insure? A game-theoretic analysis of information security games. In *Proceedings of the 2008 World Wide Web Conference (WWW'08)*, pages 209–218, Beijing, China, Apr. 2008.
13. C. Herley. So long, and no thanks for the externalities: The rational rejection of security advice by users. In *Proceedings of the New Security Paradigms Workshop (NSPW)*, pages 133–144, Oxford, UK, Sept. 2009.
14. C. Herley and D. Florêncio. Nobody sells gold for the price of silver: Dishonesty, uncertainty and the underground economy. In *Proceedings (online) of the Eighth Workshop on Economics of Information Security (WEIS)*, June 2009.
15. J. Horton, D. Rand, and R. Zeckhauser. The online laboratory: Conducting experiments in a real labor market, May 2010. Harvard Kennedy School and NBER working paper.

16. M. Jakobsson. Experimenting on Mechanical Turk: 5 How Tos. http://blogs.parc.com/blog/2009/07/experimenting-on-mechanical-turk-5-how-tos/, July 2009.
17. D. Kahneman and A. Tversky. *Choices, values and frames*. Cambridge University Press, Cambridge, UK, 2000.
18. C. Kanich, C. Kreibich, K. Levchenko, B. Enright, G. Voelker, V. Paxson, and S. Savage. Spamalytics: An empirical analysis of spam marketing conversion. In *Proceedings of the Conference on Computer and Communications Security (CCS)*, pages 3–14, Alexandria, VA, Oct. 2008.
19. A. Kittur, E. Chi, and B. Suh. Crowdsourcing User Studies with Mechanical Turk. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'08)*, pages 453–456, Florence, Italy, 2008.
20. K. Kucera, M. Plaisent, P. Bernard, and L. Maguiraga. An empirical investigation of the prevalence of spyware in internet shareware and freeware distributions. *Journal of Enterprise Information Management*, 18(6):697–708, 2005.
21. A. Matwyshyn. Penetrating the zombie collective: Spam as an international security issue. *SCRIPT-ed*, 4, 2006.
22. T. Moore, R. Clayton, and R. Anderson. The economics of online crime. *Journal of Economic Perspectives*, 23(3):3–20, Summer 2009.
23. T. Moore and B. Edelman. Measuring the perpetrators and funders of typosquatting. In *Proceedings of the Fourteenth International Conference Financial Cryptography and Data Security (FC'10)*, pages 175–191, Canary Islands, Spain, Jan. 2010.
24. Y. Namestnikov. The economics of botnets. *Analysis on Viruslist. com, Kapersky Lab*, 2009.
25. S. Peltzman. The effects of automobile safety regulation. *Journal of Political Economy*, 83(4):677–726, Aug. 1975.
26. R. Reeder and F. Arshad. SOUPS 2005. *IEEE Security & Privacy*, 3(5):47 – 50, September-October 2005.
27. J. Ross, A. Zaldivar, L. Irani, and B. Tomlinson. Who are the Turkers? Worker Demographics in Amazon Mechanical Turk. Technical Report SocialCode-2009-01, University of California, Irvine, 2009.
28. J. Rutkowska. Red pill... or how to detect VMM using (almost) one CPU instruction, Nov. 2004. http://invisiblethings.org/papers/redpill.html.
29. S. Saroiu, S. Gribble, and H. Levy. Measurement and analysis of spyware in a university environment. In *Proceedings of the 1st USENIX Symposium on Networked Systems Design & Implementation (NSDI'04)*, pages 141–153, San Francisco, CA, 2004.
30. B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna. Your botnet is my botnet: Analysis of a botnet takeover. In *Proceedings of the Conference on Computer and Communications Security (CCS)*, pages 635–647, Chicago, IL, Oct. 2009.
31. Symantec Corp. Symantec global Internet security threat report trends for 2009, Apr. 2010.
32. R. Thomas and J. Martin. The underground economy: Priceless. *;login:*, 31(6):7–16, Dec. 2006.
33. United Nations Statistics Division. Composition of macro geographical (continental) regions, geographical sub-regions, and selected economic and other groupings, April 2010. http://unstats.un.org/unsd/methods/m49/m49regin.htm.
34. C. Wilson. Botnets, cybercrime, and cyberterrorism: Vulnerabilities and policy issues for congress. *Library of Congress Washington DC Congressional Research Service*, Jan. 2008.
35. L. Zeltser. So long script kiddies. *Information Security Magazine*, 2007.