Cent, Five Cent, Ten Cent, Dollar: Hitting Botnets Where it *Really* Hurts

Richard Ford
Director, Center for Information Assurance
Florida Institute of Technology
150 W. University Blvd
Melbourne
FL 32901, USA
rford@fit.edu

Sarah Gordon
Graduate Faculty
Florida Institute of Technology
150 W. University Blvd
Melbourne
FL 32901, USA
sgordon@symantec.com

ABSTRACT

Spyware, Adware, Bots. In each case, there is significant evidence that there is an increasing financial motivation behind the writing and distribution of these programs. In this paper, the concept of using our knowledge of these financial motivators to combat malicious software is introduced. Can attacks on business models actually provide relief that technology alone cannot? Can we deploy our technology differently, in order to receive direct benefits of this indirect attack on revenue streams? Our conclusion is that not only is this a possible solution, but that it may be an extremely effective one. This is illustrated by a description of our business model attack generator, MARK – the Multihost Adware Revenue Killer. Using MARK, we demonstrate simple but effective attacks against Malicious-code generated revenue streams. However, the creation and deployment of MARK raises thorny legal and ethical questions, as the impact of the technology is widespread and could easily be targeted at legitimate online marketing models. Do the ends justify the means?

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and Protection*

General Terms

Economics, Security

Keywords

Adware, Business Models, Malicious Code, Spyware, Virus, Worm

1. INTRODUCTION

Pecuniae obediunt omnia, or so it is said. Money. Loath it or love it, a grasp of the underlying economics of a system is crucial if one wants to understand it fully. Similarly, attempting to change a system's behavior without a good overall understanding of it is pure foolishness; one has no idea if the changes are optimal for the desired outcome. Despite the inescapable logic of the need for viewing the system in the "big picture", when designing methods for combatting malware, all too often the approach taken ignores *systemic* issues and focuses purely on technological ones.

In this paper, the problem of Spyware, Adware and Botnets is examined first from an economic perspective, and only then technologically. For example, what makes Adware and "drive-by downloading" lucrative? What is the economic motivation for Spyware and Adware development? Once we understand the system holistically, we are in a position to ask if the system can be changed, and if so, can the route of change be technological in nature? Our conclusion is that such attacks are not only effective, but that they may be the most promising route for defense.

Before undertaking this discussion, it is worth spending a moment considering the different terms used, as there are significant variations in usage from person to person. For the purposes of this article, we define:

- Spyware: Programs which monitor users' activity or data, and relay this information to other computers or users, without the consent of the machine's owner.
- Adware: Programs which facilitate delivery of ad-related content to users' machines.
- Botnet: A collection of machines which are remote-controlled by some entity, without the consent of the machine owners.
- Botmaster: The person or group of people who administer the Botnet (though in this paper, we will use the term more loosely to refer to the entity controlling Adware and Spyware too).

In addition to these terms, we coin the term "Financially-Motivated Malware" (FMM) which loosely refers to a superset of the malicious code described above. While these definitions are not complete, they are sufficient for the discussions given in the remainder of this paper.

NSPW 2006, September 19-22, 2006, Schloss Dagstuhl, Germany.

Using these definitions, a brief overview is given of the economy of FMM. This economic system is then examined from a technological perspective, and the effectiveness of economic attacks on certain types of malicious code demonstrated. Finally, the limitations and complications of this approach are discussed; in particular, attention is given to the possibility of the attack being deployed against legitimate business models.

2. THE SPYWARE/ADWARE ECONOMY

Historically, most MMC (Malicious Mobile Code) development has been the electronic equivalent of joyriding. As such, studies like [4] have shown that most "old-style" virus writers "age out" of virus writing as they develop ethically. The only adults involved in virus writing activities were outside ethical norms when measured on the Kohlberg scale, and often exited the virus writing community for other reasons.

However, over the last few years, as more personal and financial information is stored on computers and as the population of computer users has both increased and broadened, there has been an increase in the observation of financial incentive for producing or deploying malicious software. Criminals have begun to use computer programs to commit facilitate the commission of crime, and some hackers and virus writers have begun to explore the fiscal possibilities available.

This evolution of the use of malware has changed some of the end-goals of the attacker. If one is attempting to "infect" machines for profit (as opposed to fun), there is a balance between rapid spread (that is, wide initial dissemination) and stealth. For example, one might suppose that the goal of a criminal is not peak spread, but the long-term longevity of that spread. Thus, virus, worms and Trojans which are disseminated quietly may achieve the largest long-term population as they remain under the antivirus radar.

Because of the long-term interest in keeping machines infected, there is a strong incentive for infected machines to "phone home" in some manner so that clients can be updated or morphed. Thus, FMM is seldom "fire and forget" but actively managed and monitored. This is often accomplished via an IRC channel (for an example of this see [7]), where infected hosts log in and await commands and/or software updates. Additionally, infected machines often download other content to infected machines. This content is usually used for revenue generation; for example, a Botnet-owned machine often has Adware installed on it, as this Adware generates revenue for the Botnet owner.

3. TECHNOLOGICAL PREVENTATIVES

Stopping MMC is difficult, as the problem is demonstrably incomputable as it can be neatly reduced to the halting problem [1]. Despite this, there has been significant effort in creating MMC detection schemes which are "good enough" to provide real-world protection. Most of these solutions are simple pattern scanners, which detect Malcode which is already well-known. These solutions can effectively reduce the risk to client machines if they are updated automatically, and if the MMC is known or similar enough to a known piece of malcode. Unfortunately, these solutions are by necessity reactive, leaving defenders continually behind the attacker. For this reason, and because of the ability for MMC to spread worldwide in a matter of minutes [8], researchers have been looking for methods of detecting new MMC in real time. One particularly promising approach is behavioral analysis, which attempts to classify processes based upon their actions [3]. Other approaches include heuristics, expert systems, integrity shells and neural networks [10].

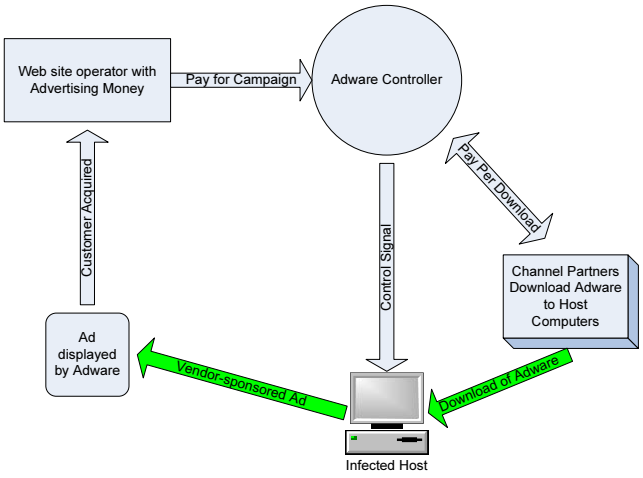

Figure 1: An overview of the different players in the online advertising space. Here, we see a highly-simplified diagram of typical relationships in the FMM world.

Overall, scanning-based solutions protect the largest number of endpoints on the network, and while there is a growing movement toward more advanced (i.e. generic) techniques, the adoption rate is fairly slow. Aside from the documented ineffectiveness of such reactive solutions for worms, such solutions are also not likely to work well against FMM, as the administrator of the Malcode has a strong incentive to update the clients regularly in order to evade detection. Instead, an uneasy equilibrium between attack and defense exists, where the level of protection scales in proportion to the perceived pain on the part of the users.

Aside from maintaining the *status quo*, the entire defense system is predicated on removing enough clients that the attacker will give up and go home. For example, consider the case of a Botnet. If our defenses are based upon cleaning the clients of the Botnet, the bot owner will simply counter this by attempting to send out more clients (preferably updated clients which are undetected by current antivirus software). Ultimately, this approach does not really "kill" the Botnet; instead, it limits the number of clients until a steady-state balance between client birth and client death is achieved. The Botnet will only truly die if the bot owner gives up on it – that is, when maintaining the Botnet becomes more trouble than it is worth.

In the preceding example, the word "trouble" is used loosely; in fact, there is a loose cost-benefit analysis. In [5], this is represented as follows:

$$M_b + P_b > O_{cp} + O_{cm} P_a P_c$$

Where M_b is the monetary benefit, P_b is the psychological benefit, O_{cp} equals the psychological cost of the activity, and $O_{cm}P_aP_c$ is the "expected penalty effect". (O_{cm} is the monetary cost/penalty, P_a the probability of arrest, and P_c the probability of conviction).

This model makes sense from the standpoint of many computer crimes – even those which are not financially motivated, such as most virus writing. More specifically, it fits well with the Botnet example we gave above. In the preceding case, the Botnet owner will give up on the Botnet when the psychological cost plus the threat/cost of conviction outweighs the psychological and monetary benefit.

Now, let us examine our current approach to protection: removing individual nodes from the Botnet. This directly decreases the monetary benefit, and one might presume that the psychological benefit also decreases as the network size diminishes – but only incrementally. The Botmaster can readily redress this by increasing the number of clients in the network, or making it harder for clients to be detected. Thus, the inequality holds true for systems which are just limited by virus scanners; the probability of conviction is low, and the "cat and mouse" nature of the relationship with the anti-virus world can be quite satisfying. Finally, the returns on the actions can be quite large. Because of this, while our current technological approaches to prevention of MMC have some effect, they do not really impact the cost-benefit calculation.

Re-examining the equation given above, it is clear that more effective ways of impacting the revenue stream are required – that is, we need to decrease the viability of the business model. As touched on previously, current efforts seek to limit the size of the Botnet, thereby driving up the cost of acquisition and maintenance of the net. However, the revenue model for Botnets is rather more complex, and bears further examination.

In general terms, a Botmaster makes money in one of three different ways[1]:

- Advertising/Software Install
- Theft and exploitation of confidential data
- Distributed Denial of Service (DDoS) for hire.

While verifiable data are difficult to gather due to the underground nature of the FMM world, first-hand empirical evidence points toward the first list item as being the primary revenue source for most Botnets, and so it is this source upon which we shall focus.

3.1 The FMM Advertising Revenue Model

Online advertising has changed dramatically from the early days of the Internet. Essentially, there are three primary revenue models which support ad hosters[2]:

1. **Pay per impression.** The website or software which displays an advertisement is paid a small sum of money for each impression of the ad. This system can be monitored by the advertiser fairly simply, as usually the ad is hosted on the advertiser's own system. Thus, each time the ad is viewed, a log entry is made. This allows both the buyer and the ad hoster to verify that the number of impressions paid for is correct.

2. **Pay per click.** One of the drawbacks with pay per impression advertising is that there is little control on how an ad is displayed. For example, an unscrupulous ad hoster could display many hundreds of ads on a single webpage, driving up revenue. However, the benefit to the advertiser would be very limited, as the ad would essentially be lost on the page – that chances of the ad being noticed by a potential customer are low. A better model is pay per click, where ad hosters are paid for the traffic they drive to a particular website. Once again, in this model both parties can independently verify the number of clickthroughs generated by a campaign.

3. **Pay per sale.** This model usually pays the highest amount to the ad hoster, as the conversion rate (that is, sales per advertisement impression) tends to be very low. However, the model is attractive in that the advertiser only pays for "successful" ads, and as such, can afford to pay significantly money for each customer acquired. If the ad hoster has a good customer base, and displays the ads in such a way as to acquire a high click through rate, this model can be highly lucrative. The drawback is that it is difficult for both parties to independently verify the number of resulting sales. As such, it is not as common in the advertising underworld as the other models described above.

Each of these revenue models has a place in the legitimate world of online advertising: they exist for banner ads, pop-ups, ad driven software, and email. However, in addition to the legitimate uses of these techniques, they are also extremely popular in the botnet/adware world. An adaption of these models is where an ad platform vendor (generally, an Adware developer) is prepared to pay distributors for every new client the Adware is installed on.

The preceding paragraph touches on another crucial component of the revenue model: the existence of distributors. There is little economy of scale for individual website operators contacting advertisers individually; instead, clearinghouses for online advertising, like Doubleclick, sell advertising space which is displayed throughout its large reseller network [2]. Thus, a small website can display banner ads through the Doubleclick system. Doubleclick is paid by the advertiser, and in turn pays its resellers, who provide web

[1]It is tempting to include a fourth item in this list: renting the botnet out to someone else. However, in general the renter herself has a goal of one of the three listed money-making schemes.

[2]Such models are in a constant state of flux, see: http://digitalenterprise.org/models/models.html for a discussion of online business models.

space and traffic for the Doubleclick system. The relationship is entirely synergistic, and, when running properly, benefits all members of the system, including those who view the advertisements. However, the corollary is that the system is predicated on each member of the system receiving good "value for money" (or, the case of the user, value for time). Further, the financial relationships are governed by contracts – these contracts are a fixed point which cannot easily be changed in real time. This observation opens the door to a number of interesting attacks.

4. CONTROLLING WHAT YOU CONTROL

When viewing FMM from this business model perspective, novel solutions present themselves. For example, if the cost-benefit equation of the revenue models given are correct, how else might we use technology to affect either the cost or the benefit to the attacker?

The most important thing to recognize is that users have a very good control of certain parts of the system. In fact, the client machine controls how often an ad is displayed, how often it is clicked upon, and even how many purchases or signups are made to a system, or how often a client is downloaded.

With this in mind, let us conduct a *Gedanken* experiment. Assume, for a moment, that we have a Botnet owner who is installing Adware on "0wned" machines. This Adware in turn displays ads paid for by by a fictional adgroup we shall call "Ad Consortium". In terms of business model attacks, the obvious one is to detect and remove the clients from the infected machines. This reduces the revenue of the Botmaster, but the Ad Consortium is still happy as their costs are also lowered. However, what happens if the number of impressions goes *up*?

In a pay-per-impression model, the Botmaster earns more money as the number of impressions increases. If this is due to more users actually seeing the ads, then the Ad Consortium (and, indirectly, its customers) are happy, as the campaign will generate more web traffic. However, what if the impression count goes up artificially? That is, what if a third party was deliberately downloading ads, but not displaying them? These extra downloads would generate more revenue for the Botmaster, but the Ad Consortium would not be happy: it pays per impression, and so would be spending more money for no increase in returns. Over a period of time, it is very likely that the per impression fee would drop, as the Ad Consortium payouts are ultimately linked to the actual sales that each impression leads to. Ultimately, then, a new steady state would ensue, where the actual number of sales through the Ad Consortium is unchanged, but the number of impressions is higher (with a lower fee paid per impression).

Creating such a change in impressions is actually quite easy. One could use Virtual Machines, for example, to become virtual Botnet members. These virtual machines would drive impressions of ads via the installed Adware, but would never make a purchase. After a period of time of increased activity, these virtual machines are suddenly disabled; this causes a precipitous drop in ad impressions. However, each impression pays less than it did before; the Botmaster therefore suffers a large drop in revenue. Over a period of time, the cycle of increased impression followed by a precipitous drop repeats itself.

From a business perspective, this causes a number of different problems for each member of the community. A simplified diagram of these relationships is shown in Figure 1.

For the Botmaster, revenue will initially increase but then suddenly decrease. For the Ad vendor, customers will be upset as advertising expenditure will go up with no increase in sales.

Using this approach, the revenue generated by the Botmaster is directly effected. Furthermore, there are other important business realities. For example, contracts generally have a particular duration. That is, the fee per impression is generally set for a period of time. Also, in real campaigns, there are often fluctuations in click through rates, so it will take time for a vendor to realize that conversion rates are dropping. This leads to a lag between systemic response and the stimulus. This lag could also be used to make the business model untenable.

This attack is represented graphically in Figure 2. At the beginning of the graph, the system is in a simple equilibrium; furthermore, we assume each member of the group is making enough money that the system is profitable for every participant (except, of course, the unwitting host of the FMM). We now use a distributed group of clients to ramp up the number of impressions slowly. This has the effect of driving up the costs of the advertisers, and increasing the revenue of the Botmaster. As the ads are no longer being as effective, the large arrows represent renegotiation of the "per impression" fee by the advertiser. The system once again settles into equilibrium. The rate of "fake" impressions is once again increased, cutting into the advertiser's revenue (by increasing costs with no increase in revenue). This cycle can be repeated ad-infinitum, until the advertiser no longer wishes to do business with the botmaster.

In this example, our conclusion is based upon our understanding of the adware-driven revenue model. As previously discussed, advertisers can easily trace impressions and clickthroughs to particular advertising campaigns. Thus, this revenue model is extremely popular as it requires no bilateral trust between the ad-hoster and the advertiser: each party can tally impressions and determine what the actual impression count was. In online advertising, the *placement* of advertisements is important. For this reason, savvy advertisers generally monitor not only the display rate, but the clickthrough and conversion rate (that is, how many ads actually turn into customer acquisitions). Advertisers are willing to pay more for adspace which brings in customers not just page views. Thus, our attack essentially "devalues" the space sold by attacked providers. Once this new value becomes the market norm, the additional (and artificial) download rate can be removed; this would have the net result of leaving the vendor with lower income than prior to the attack.

It is possible to illustrate the efficacy of this attack by using some "back of the envelope" calculations. Let C_i be the

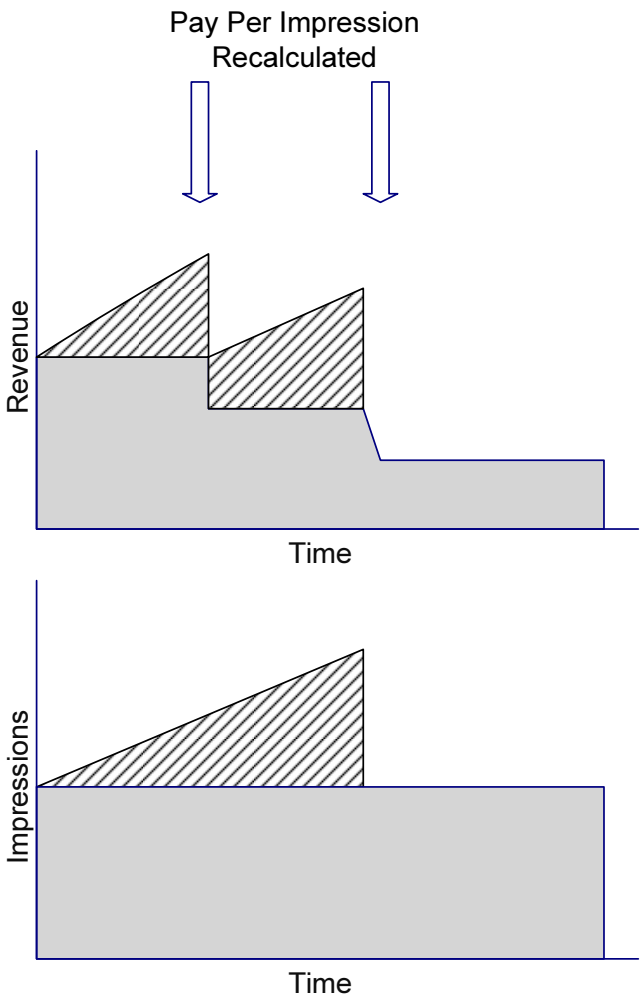

Figure 2: The lower graph shows the number of impressions as a function of time, as our network of collaborating hosts increases traffic. The hatched region in the upper graph shows the extra revenue generated by the botmaster. At each arrow, the advertiser adjusts the payment per impression based upon profitability and conversion rate.

payment per impression the advertiser is willing to make. The ad campaign makes financial sense to the advertiser when the monies spent on advertising equal the profit made from sales. If we define R_c to be the conversion rate of ads to sales, P to be the profit made per sale, and N to be the total number of ads displayed, we can find the break-even point where money in is equal to money out:

$$C_i \times N = R_c \times P \times N \quad (1)$$

$$C_i = R_c \times P \quad (2)$$

However, N is not necessarily a "real" representation of actual impressions. If N is considered to be composed of I_r and I_f where these symbols are the real and fake (spurious) impressions, the equation above changes, as only I_r can generate sales. Thus, we have:

$$C_i \times (I_r + I_f) = R_c \times P \times I_r \quad (3)$$

$$C_i = \frac{R_c \times P \times I_r}{I_r + I_f} \quad (4)$$

Thus, we can see that as I_f is varied, the Cost per impression for breakeven changes. In fact, as I_f increases, the advertiser can afford to pay less per impression, as sales do not increase. Illustrating this with real numbers, let us assume a conversion rate of 1%. If the average profit per sale was $10, this gives a value of $0.10 per impression. As the ration of I_r to $I_r + I_f$ changes, this value will drop, reaching $0.05 per impression when $I_r = I_f$.

Once this new equality takes hold, I_f can be altered once more; the net result of this is that *predictability* is taken out of the equation. The simple act of reducing I_f reduces the adware vendors revenue. Similarly, randomly changing I_f changes the conversion rate (the ratio of impressions to sales) – something which is very suspicious to the advertiser, as the practice of generating "phantom" clicks is not unheard of within the industry.

Consider another example. Some Adware vendors pay "resellers" a bounty for each install of the Adware on a new machine. This bounty can be quite substantial - that is, of order dollars, not cents. Resellers are therefore highly motivated to install the Adware on user's machines. Often this is done legitimately ("Download this useful client and use it. In return, we'll display advertisements on your machine"). Often it is not.

If we once again construct an attack using virtual machines, it is possible to drive a very large number of downloads of the Adware, costing the Adware vendor massive sums of money. Furthermore, both the Adware vendor and the reseller can independently count the number of the downloads, making it difficult for the Adware vendor to back out of payment legitimately. After a suitable waiting period, the VM's hosting the Adware are replaced. The net result of this is that the resellers of the Adware benefit, but the actual source of the money is injured. If this attack were to be repeated, it would make this method of Adware distribution financially unworkable.

Once again, the structure of the Adware world makes this approach viable. Our belief is that Adware vendors do not

generally have large reach into the world as they lack their own independent method for driving content. Instead, they rely on a network of redistributors who have either large mailing lists (usually, spammers) or high-volume websites. These resellers then make money per Adware client installed through their media. Adware vendors simply do not generally have access to the sort of high-volume media outlets required to disseminate their products. Keeping this reseller chain happy means providing competitive payouts per impression or per email. Thus, many Adware vendors are banking on future sales of advertising space to pay for the installation of clients now. This "bet on the future" leaves vendors open to a rather unpleasant attack.

5. SYSTEM DESIGN: MARK

The attacks described above could be carried out quite easily using a distributed network of host computers. This system "Multihost Adware Revenue Killer" (MARK) would be simple to build using readily available components. For example, most machines are capable of executing a virtual machine (for example, VMWare or Virtual PC) which can safely host various adware/spyware/botnet clients. These machines could be run at night when the systems were idle.

Using these clients worldwide, it would be trivial to launch revenue-destabilization attacks on any deployed network of compromised machines. Furthermore, some of the machines could be tasked with crawling the web, looking for botnets/adware to become infected by. Based upon studies [9], acquiring adware/spyware in this manner should be rather straightforward; additionally, longitudinal analysis of the data should be able to point us to the largest networks so that the technique provides the best possible return on investment: the largest (working on the assumption that breadth of distribution channel is loosely proportional to the ultimate size of the network) net can be attacked first.

Once a network has been targeted, MARK nodes will download client software from as many sources as possible. This software will execute on a virtual machine, so that there is little or no danger to the host operating system. This download already perturbs the revenue model for the target as many of these locations are likely to be hosted by resellers: thus, pay per clickthrough or pay per download fees will be accrued at this point in the attack.

Next, the infected systems will begin to execute the downloaded code. Once again, this further modifies the revenue models, increasing the profitability of some actors in the system, and decreasing that of others.

Actions taken at this point will depend upon the actions of the code downloaded. If, for example, the code appears to be Spyware, continued downloads and installs from as many clients as possible may be the most effective route, as the Spyware vendor is likely to be paying resellers for each successful install. If the code is simply Adware, the best attack may be to gradually increase the numbers of ad impressions and clickthroughs. This will initially increase the revenue for the Adware vendor, but will likely sour relationships with the advertisers themselves, as their expenditure will be increasing with no corresponding increase in sales. Furthermore, it would be difficult to distinguish between MARK-generated traffic and click-fraud on the part of the Botmaster.

Finally, after the attack is complete (indicators of effectiveness might be a decrease in members of IRC control channels, or a reduction in sites hosting the "bait" for the malicious code), the Virtual Machines (VMs) are refreshed to a known-clean state and another target is located (and, hopefully, destroyed).

VM detection is possible as a countermeasure, but it would suffer from a number of limitations. First, it is difficult to detect the presence of a virtual machine - in the case of a hypervisor, some have even claimed it is impossible (see for example the media hype surrounding "Blue Pill"). Second, the manager of the botnet/adware/spyware actively *wants* the software to be installed – the goal is to build a larger network of machines as this will drive more revenue. Thus, a Botmaster who wants *many* clients has a problem: actions which make it more difficult for MARK to find, install and monitor Adware will also decrease the rate of real users who become infected. Thus, it is difficult for the Botmaster to prevent MARK obtaining samples without cutting down on "real" revenue opportunities. Similarly, if the traffic generated by MARK clients is built slowly, it is not obvious which clients are real and which are simulated, making it difficult to cull fake clients. Being too picky about the nature of clients will naturally cut down on the potential install base.

The preceding discussion is predicated upon an important assumption: that the botnet world is a numbers game, where the larger the network the more ability the Botmaster has to make money. It is certainly possible to construct scenarios where this is *not* the case, however, those examples represent a fundamentally different revenue model. As such, for the purposes of this paper we assume that the desire of the Botmaster is for a larger network.

6. LIMITATIONS

This revenue-based technique basically relies on the idea of driving the system to a new – but artificial – equilibrium, which can be broken at any time by the removal of our artificial driving force. The question is, would constructing such a system be legal? Even if such a system is currently legal, is it ethical?

The legality of the approach may ultimately be governed by the End User Licensing Agreements of the clients downloaded. It is entirely possible, for example, to conceal terms in the EULA which strictly forbid the software's use in the manner outlined in this paper. However, this prophylactic is not likely to be effective, as adware/botnet groups are reluctant to emerge from the underground community and press legal charges.

A much more difficult question is whether the approach is ethical.

The question here hinges on the ability for the system to be massively abused. Click fraud is already a large issue for online advertising companies [6]; the MARK framework could easily be leveraged to instigate this abuse on a hitherto unthoughtoff scale. Choosing to deploy a system which would

completely destroy a large section of the online economy is a decision which requires serious contemplation.

There is certainly a temptation to argue that online advertising is evil and if it ends up as the unwitting victim of MARK, too bad. However, this argument is *highly* simplistic. Most users fail to recognize that distributing high-quality content online costs money; this money either comes from subscriptions, or, more commonly, advertising revenue. Google is the perfect example of this: imagine a world where the only search engines are "for fee". Thus, the total destruction of online advertising is not a desirable outcome of the development of MARK.

Similarly, one might argue that pay per click systems are already trivial to defraud; a system as sophisticated as MARK is completely unnecessary. While there is an element of truth to this, it seems that most attackers are still fairly simplistic.

In the preceding discussion we have focused primarily on advertising-related revenue models; however, FMM can be used for far more sinister purposes: spying on users, and launching attacks on other machines. The effectiveness of business-model related attacks is different in each case. When gathering confidential data on users (such as passwords, usernames and account numbers), the utility of the data relies on its accuracy. That is, there are some risks involved with using the gathered data. If a large amount of the data were incorrect, it is possible that the value of that data would be reduced. Furthermore, correctly seeded, such data could be used as an investigative tool to increase the probability of conviction of those who use the data.

Leveraging the business model of the Botmaster when considering DDoS attacks is more troublesome. Here, the utility of the attack relies upon simply having a large number of machines overload the hapless victim site. Payment is presumably predicated upon the target site(s) actually being disabled. Currently, there is no obvious way to attack this business model directly, as the most important elements seem to be beyond direct control of the client. However, if this were to be the only revenue stream available to Botnet owners it seems likely that many of the smaller Botnets would be economically unviable. Furthermore, psychologically, downloading Adware is seen as "less wrong" that a direct attack on another website and extortion.

Finally, there is one limitation which is difficult to address: it is possible that MARK simply will not work, due to our limited understanding of FMM revenue models. However, it is our firm belief that injecting unpredictability into the delicate financial equilibrium online is likely to have some effect on the robustness of the models. Either way, the only way to deal with this criticism is to actually deploy MARK and measure its results (though even this measurement process is difficult).

7. CONCLUSION

It is crucial to understand the changes financial motivation bring to possible methods to reduce Adware and Botnets. In particular, the steady-state equilibrium required within the advertising system allows us to deliberately destabilize Adware business processes by modifying only factors which defenders can influence directly. By constructing a distributed network of machines capable of controlling advertising impression numbers, click through rates and software package installs, it should be possible to dramatically change the economics of online advertising.

It is our belief that the approach outlined here would be highly effective at reducing advertising revenues generated by both Adware and Botnets. Given that this is an important component in many such networks, it seems plausible that an organizing campaign directed at different Botnets could convince some Botmasters to simply give up maintenance of the network.

Unfortunately, the technology as described here is double-edged in that it is equally effective at attacking legitimate business models. However, it may be possible to significantly limit if not eliminate this undesirable side-effect. In any case, our belief is that this side-effect is significant, but not necessarily overwhelming. Online advertisers need to understand click fraud more completely, and deploy systems which are more directly related to revenue; this should be possible in the legitimate world, and impossible in the underground world of the Botnet.

As computer crimes continue to have a strong financial component, disregarding opportunities to put the bad guys capital at risk is foolishness. Every method for making computer crime less attractive should be explored. MARK is an obvious weapon, which, when wielded intelligently, could make a huge difference in the battle.

8. ACKNOWLEDGEMENTS

Dr. Ford would like to thank David Ladd from Microsoft Research for sponsoring the work outlined in this paper.

9. REFERENCES

[1] F. B. Cohen. *A Short Course in Computer Viruses*. John Wiley & Sons, New York, 2^{nd} edition, 1994.

[2] Doubleclick. Doubleclick: Digital advertising, 2006. Downloaded April 2006 from `http://www.doubleclick.com/us/`.

[3] R. Ford, M. Wagner, and J. Michalske. Gatekeeper II: New approaches to generic virus prevention. In *Proceedings of the* 14^{th} *International Virus Bulletin Conference*, Chicago, IL, 2004. Virus Bulletin.

[4] S. Gordon. The generic virus writer II. In *Proceedings of the* 6^{th} *International Virus Bulletin Conference*, Brighton, UK, 1996. Virus Bulletin.

[5] N. Kshetri. The simple economics of cybercrimes. *IEEE Security & Privacy*, 4:33–39, 2006.

[6] C. C. Mann. How click fraud could swallow the Internet. *Wired*, 14.01, 2006.

[7] B. McCarty. Botnets: Big and bigger. *IEEE Security & Privacy*, 1(4):87–89, 2003.

[8] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the Slammer worm. *IEEE Security and Privacy*, 1(4):33–39, 2003.

[9] A. Moshchuk, T. Bragin, S. D. Gribble, and H. M. Levy. A crawler-based study of Spyware on the Web. In *Proceedings of the* 13^{th} *Annual Network and Distributed System Security Symposium*, San Diego, CA, 2006. (NDSS 2006).

[10] P. Szor. *The Art of Computer Virus Research.* Symantec Press, 2005.